**CAPSTONE PROJECT**

**MICROSERVICE BOOTCAMP**

Microservices Certification Training – Edureka

**VAIDEESWARAN K**

**1876622 - TCS**

**E-COMMERCE - MICROSERVICE ARCHITECTURE**

**Diagram, schematic

Description automatically generated**

**Zuul API Gateway:**

It works as a front door for all requests, it handles all the incoming request and performs dynamic routing of microservice applications. Based on the information from Eureka it gets the available services and connects the respective microservices with the service name. Zuul API Gateway runs in the default port 127.0.0.1:8080.

**Eureka:**

It holds the information about all client service application running on each port. Every microservice registered to this Eureka service by providing the URL. Eureka runs in the default port 127.0.0.1:8761.

For this project there are three main services Orders, Products and Shipping and rest two services Products Recommendation and Shipping Consumer/Deliver Notification services performs based on information from the main services.

Load balancer is implemented in the services. So, when any services is down it will redirects traffic to the available instances.

Graphical user interface

Description automatically generated

**Products Service:**

Table

Description automatically generated

It has some predefined values in h2 database. Products service having information about product ID, description, product category, price, and quantity. Product service has the following methods,

**createProductsDetails** – insert new products to the database

**Request:** (POST) <http://localhost:8080/products/products>

Graphical user interface, text, application, email

Description automatically generated

**deleteProductDetails** – delete the products based on id.

**Request:** (DELETE) <http://localhost:8080/products/products/5>

Graphical user interface, text, application

Description automatically generated

**getmatchedproducts** - to get products based on either value of Description or Category.

If parameter values not mentioned, it will filter all the records from table.

**Request:**

(GET) <http://localhost:8080/products/products?description=iPhone13>

(GET) <http://localhost:8080/products/products?category=phone>

(GET) <http://localhost:8080/products/products>

**getproductsbyproductid** - to get products based on product id.

**Request:**

(GET) <http://localhost:8080/products/products/100>

Graphical user interface, text, email

Description automatically generated

**patchProductDetails –** It will update only particular item in the table.

**Request:** (PATCH) <http://localhost:8080/products/products/100/8>

Graphical user interface, text, application

Description automatically generated

**updateproductsdetails –** It will update all the items in the table.

**Request:** (PUT) <http://localhost:8080/products/products/101>

Graphical user interface, text, application, email

Description automatically generated

**Order Service:**

Graphical user interface, text, application, email

Description automatically generated

It has some predefined values in h2 database. Order service storing information about the ordered products with order id. When any order is initiated, it will check the stock is available from products services with REST API calls and it will throw error message if there is any out of stock. Or else it will process the order request and share that customer address information to the shipping service with order status using REST API calls. Finally, it will update the remaining stock quantity in the products service.

This order service having one PUB/SUB calls with Product recommendation service where it will share only the product category and product description.

Circuit Breaker and Hystrix dashboard is enabled, so it will monitor the incoming request and shows the hits and status of the circuit whether it is closed, open or half-opened state.

**createordersandcallshippingservice** – This method stores the order information and using the Rest Template it sends the shipping details with total price to the shipping service. Using the Producer/Consumer with ActiveMQ sends the order information to products recommendations service.

**Request:** (POST) <http://localhost:8080/orders/orders>

**Request JSON Body**: {

  "orders":[

    {

        "id": 1,

        "productID": 100,

        "description": "iPhone13",

        "category": "phone",

        "price": 999.99,

        "quantity": 2

    },    {

        "id": 2,

        "productID": 101,

        "description": "series 7",

        "category": "iwatch",

        "price": 599.99,

        "quantity": 3

    }    ],

  "shipping":{

    "name":"vaidees",

    "phone":"0123456789",

    "address":"Chennai",

    "status":"Order Received"

  }

}

**Case 1 – When Stock is Available**

Graphical user interface, text, application, email

Description automatically generated

**Case 2 – When there is Out of Stock**

Graphical user interface, text, application, email

Description automatically generated

**getallorders** – Get all Order details from the table.

**Request:** (GET) <http://localhost:8080/orders/orders>

Graphical user interface, text, email

Description automatically generated

**Hystrix Dashboard** – Circuit breaker is enabled to monitor if any fails when calling the shipping or products service

**Request:** (GET) <http://localhost:6001/hystrix>

Graphical user interface

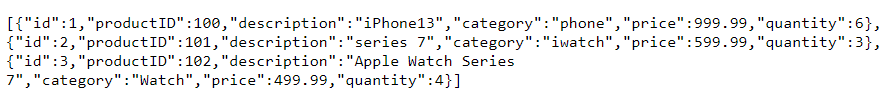
Description automatically generated with medium confidence

**1.RestTemplate - Calling Products Service from Order Service:** (POST) <http://localhost:8080/orders/orders> REST API call hit the order service and it calls the products service to check current stock availability and once order created it will again call products service to patch the products with new order stock quantity.

**Request:**

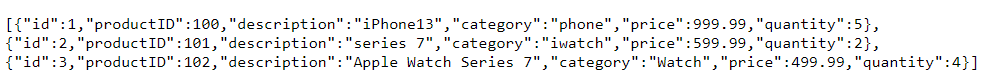
(PATCH) <http://localhost:8080/products/products/101/1>

(PATCH) <http://localhost:8080/products/products/100/1>



Graphical user interface, text, application, email

Description automatically generated



**2.RestTemplate - Calling Shipping Service for Order Service:** Order service have the information about the customer shipping address, so it calls the shipping service and sends the shipping information with orderID.

**Request:** (POST) <http://localhost:8080/shipping/shipping>

**Request JSON Body**:

{

    "name":"vaidees",

    "phone":"0123456789",

    "address":"Chennai",

    "status":"Order Received"

  }

**ActiveMQ – jmsTemplate –** Order service sends the product category and description to the product recommendation service for products recommendation purpose.

**convertAndSend to Products Recommendation Service –**

{

    "category":"phone",

    "description":"iphone13",

}

**Shipping Service:**

Graphical user interface, application

Description automatically generated

It has some predefined values in h2 database. Shipping service storing information about the order status and customer address information. Once shipping information is added it sends the orderID, shippingID, order total, customer name and phone number to the Shipping Consumer Service/Notification Delivery service using the PUB/SUB calls.

Order status can be changed from Order Received to Shipped or Delivered status with the patch request calls.

**createshipment –** Order service will call this shipping service using resttemplate and send the customer order status and address information. This information will be inserted into shipment details.

**Request:** (POST) <http://localhost:8080/shipping/shipping>

**getmatchedshipping –** To get shipping details based on either value of ShippingID or OrderID. If parameter values not mentioned, it will filter all the records from table.

**Request:**

(GET) <http://localhost:8080/shipping/shipping?orderid=84>

(GET) <http://localhost:8080/shipping/shipping?shippingid=3>

(GET) <http://localhost:8080/shipping/shipping>

Graphical user interface, text, application, email

Description automatically generated

**patchshippingDetails –** It will change the order status field if there is any change in the order delivery status.

**Request:** (PATCH) <http://localhost:8080/shipping/shipping/5>

Graphical user interface, text, application, email

Description automatically generated

**ActiveMQ – jmsTemplate –** Shipping service will send the orderID, shippingID, order total, customer name and phone number information to the Shipping Consumer Service/Notification Delivery service.

**convertAndSend to shipping consumer service –**

{

" Name":"phone",

" Phone":"0123456789",

" OrderID ":"85",

" shippingid ":"6",

" Total ":"1599.98",

}

**Products Recommendation Service:**

Application

Description automatically generated with medium confidence

It has some predefined values in h2 database. This service having the product category and description information. When order created in order service, this products recommendation service consumes that product information and stored the details.

**getallproductsforrecommendation –** Get all products category and description from recommendations list.

**Request:** (GET) <http://localhost:8080/productsrecommendation/productsrecommendation>

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

**Order Service to Products Recommendation Service:**

Graphical user interface, text, application, email

Description automatically generated

**Shipping Consumer/Notification Delivery Service:**

This service has the information about orderID, shippingID, order total, customer name and phone number and this service will deliver notification to the customers.

**Shipping Service to Shipping Consumer/Notification Delivery Service:**

Graphical user interface, text, application

Description automatically generated

**ELK Logging:**

Each service having the following log files linked to logstash. And distributed tracking is implemented using the zipkin and sleuth.



**LOG FILES:**

**Products Service:** products-log.log

**Order Service:** orders-log.log

**Product Recommendation:** productrecommendation-log.log

**Shipping Service:** shipping-log.log

**Shipping Customer/Notification Delivery Service:** shippingconsumer-log.log

**Screenshots:**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated