gRPC-Gradle-MongoDB-AWS Application

Abishek Arumugam Thiruselvi(40218896)
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
abishekarumugam@icloud.com

Srinidhi Honakere Srinivas(40221128)
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
srinidhihs007@gmail.com

***Abstract*— This Assignment aims to perform asynchronous communication for the gRPC application.**

Keywords—AmazonMQ, RabbitMQ, Gradle, Spring Framework, gRPC and MongoDB.

I. INTRODUCTION

Assignment 2 was to establish an RPC communication, MongoDB data operations, and data processing aggregate pipelines. Now in Assignment 3 To better understand the trends and patterns in these costs, this project focuses on implementing a message-oriented architecture for processing college education cost statistics. The system relies on the producer-consumer pattern, using Amazon's RabbitMQ message broker for communication. By adopting this architecture, the system allows for efficient and flexible querying and processing of education cost data.

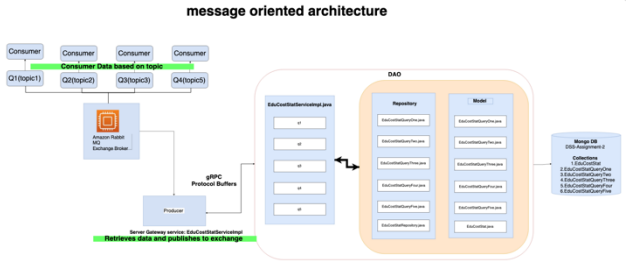

Fig. 1.1. Data Operation and Message-Oriented Architecture with RPC Communication

II. APPLICATION DESIGN

The assignment is done using Gradle instead of maven as a build tool and instead of pom.xml, we can find dependencies in build. Gradle

Fig. 1.1. build. Gradle

The Application has a source folder src/main/java and a resource folder src/test/resources.

A. dss-assignment-3: src/main/java

This source folder has 6 packages as shown in Figure Fig. 2.1. The details of the package are described below.

1. *Package: cu. dssassignment2.controller*

   This package has the file "Controller.java". It has the RestController which use to upload the dataset to MongoDB.

2. *Package: cu. dssassignment2.model*

   This package has the file "EduCostStat.java", "EduCostStatOne.java" etc. It is the class object which has all the details about data models.

3. *Package: cu. dssassignment2.utils*

   This package has the file "DssAssignment2Application.java". It is the start of the spring boot application. Here we have started the gRPC server using @Bean annotations.

4. *Package: cu. dssassignment2.repository*

   This package has all the collections interface to perform queries and save in appropriate collections.

5. *Package: cu. dssassignment2.grpc.client*

   This package has RabbitMQPublisher, which converts and sends the requests in binary to the corresponding topic exchange and its appropriate bindings/queue.

6. *Package: cu. dssassignment2.grpc.server*

   This package has service implementation gRPC which is like REST Controller. Which receives the client requests and sends the response.

B. rabbitlistener: src/main/java

RabbitMQConfig is the listener/consumer, that gets the published byte array to the corresponding topic exchange.

C. src/main/proto

1. *eduCostStat.proto*

   This file has all the proto 5 services which are required to generate service gRPC.

D. src/main/resources

The source folder contains application.properties to connect with MongoDB and elastic beanstalk.

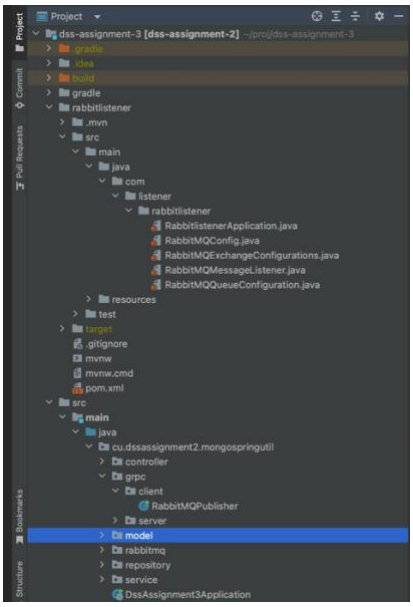

Fig. 2. 1. Project Structure

E. Mongodb cluster:

Initially, we created a mongodb cluster to which we connected to the Java application using a host link. We used this host link in application properties.

F. Mongodb Collection:

We created the mongodb collection we wrote a method in the eduCostStat controller to insert it into the collection and also make sure there is a duplicate collection that is not added when the controller runs again.

G. Models:

We have created Educoststat, EducoststatQueryOne, EducoststatQueryTwo, EducoststatQueryThree etc. making sure to have getters and setters.

H. Repositories:

EducostRepository and all other repository is where all the queries and rules are written making most part of the data access object

The RPCs that the client can call, Query1 to Query 5 are all included in the EduCostStatService. One or more protobuf messages may be used by each RPC for input and output.

From EducoststatService the data is sent/received from mongodb through the repository and model.

I. RabbitMQ -Producer:

The producer retrieves the datasets from each collection from the MongoDB cloud service for each topic listed.

J. RabbitMQ-Exchange

The producer sends the data to the RabbitMQ exchange, which then routes the messages to the appropriate queues based on the routing key.

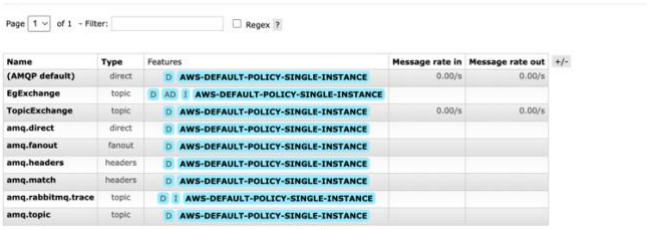

Fig. 2.2. RabbitMQ Exchange

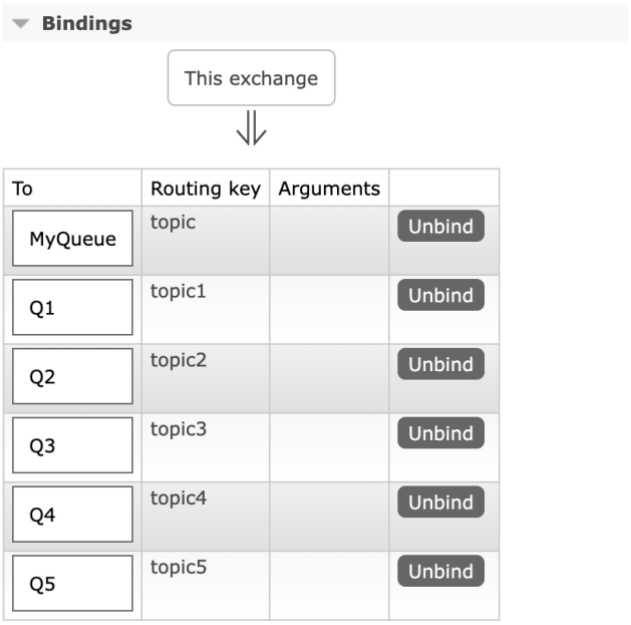

Fig. 2.3. RabbitMQ Bindings

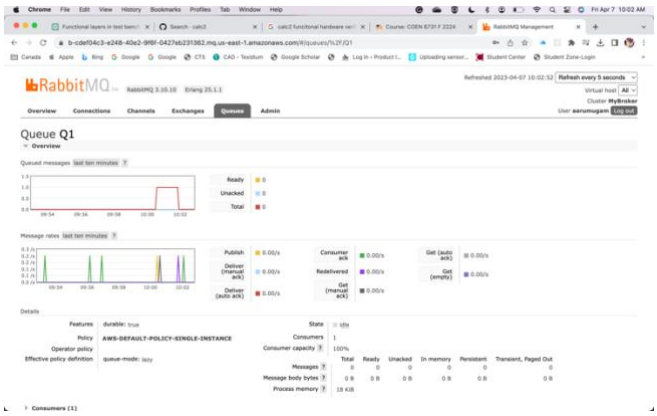

Fig. 2.4. RabbitMQ Queues

K. RabbitMQ-Consumer:

RabbitMQConfig is the listener/consumer, that gets the published byte array to the corresponding topic exchange.

III. TASKS

A. Task 1 & 2-Producer retrieves and Publish

Fig 3.1. shows the published byte array in the corresponding queue with its routing key and exchange.

Fig. 3.1. Publishing the message

B. *Task 3- The consumer listens for the message*

Fig.3.2. shows the consumer receiving the data on a separate terminal in the same node.

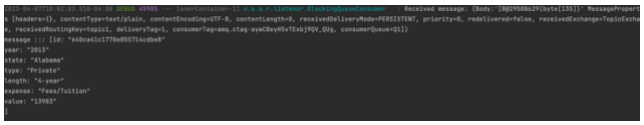

Fig. 3.2. consumer in terminal

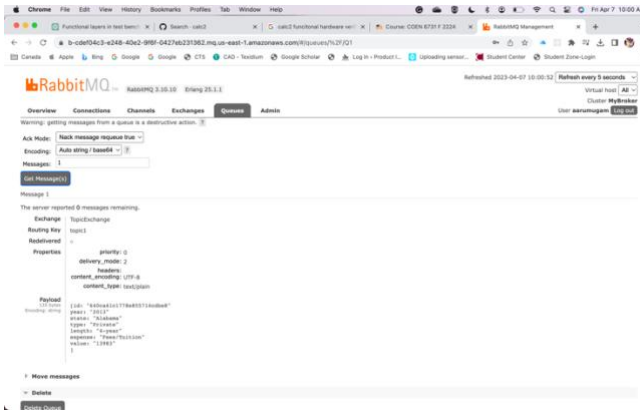

Fig. 3.3. listener in RabbitMQ

C. *Task 4- Using AmazonMQ to access RabbitMQ*

Fig. 3.3. url suggests the RabbitMQ used is from AWS. The credentials to verify are given below,

URL: AmazonMQ

Username: aarumugam

Password: guest1234567

D. *Task 5- Producer & Consumer run on the same node*

Two terminals need to be used to check the application functionality. Please follow the steps in the readme file.

IV. RESULTS

1. The producer retrieves the dataset from the collection.
2. The producer publishes the data to exchange topics with the routing key.
3. The consumer receives the data based on subscribed queue.
4. RabbitMQ is installed in AWS and also locally.
5. Producer and consumer runs on the same node and not in multithreading.

V. CONCLUSION

To run the application, review the readme file.

In conclusion, the COEN 6731 Winter 2023 Assignment Three has provided us with the opportunity to practice RabbitMQ messaging service, RPC communication, data operations on MongoDB, and the implementation of data pipelines.

REFERENCES

[1] https://moodle.concordia.ca/moodle/mod/lti/view.php?id=3433629

[2] https://github.com/abishekat/dss-assignment-3

[3] Message-Oriented Architecture has given below.