**CS6650 Assignment 3**

**Siyue Li**

1. **GitHub Repo**
2. **Description of your server design, include major classes, packages, relationships, how messages get sent/received, etc.**

The server architecture I chose for this assignment is Tomcat/Java Servlets and Amazon RDS MySQL database. Aside from the servlet itself, since we are utilizing RabbitMQ for asynchronous messaging for review api, I added another RabbitMQ server to process and consume the message queue.

**Major classes and database**

1. **AlbumServlet**

AlbumServlet is mainly designed for handling incoming POST requests related to album creation. This servlet manages the reception and processing of requests, and the requests are processed synchronously. AlbumServlet will process and validate the image file and album profile information, generate a unique albumID for each album and put each post call into the database.

1. **ReviewServlet**

ReviewServlet is to handle incoming POST requests related to album reviews. Calls to this servlet are processed asynchronously with the use of RabbitMQ. Before processing requests, this servlet will establish a connection to a RabbitMQ message broker. Then it will process the incoming requests by extracting albumID and reviewType from the url and then constructs a JSON message containing this data and publish it to RabbitMQ channel. When the message is consumed, this will also be updated to the SQL database.

1. **DatabasePool**

I choose to use AWS RDS MySQL database. The process of creating database and table is like the following:

CREATE DATABASE album\_store;

USE album\_store;

CREATE TABLE albumInfo (

AlbumID VARCHAR(255) PRIMARY KEY,

ImageData LONGBLOB,

AlbumProfile VARCHAR(255),

NumberOfLikes INT DEFAULT NULL,

NumberOfDislikes INT DEFAULT NULL

);

In summary, the schema of my database table only contains one table. Album id is auto incremented by one each time we post item.

For image, I used the image given by the professor, which is 3KB. I stored the image in the form of LONGBLOB. Blob storage is a type of cloud storage for unstructured data. A "blob," which is short for Binary Large Object, is a mass of data in binary form that does not necessarily conform to any file format.

For this DatabasePool class, a Hikari connection pool is initialized in the constructor, providing an efficient way to handle connections. The class also configures url, database username and password and connection pool parameters. It encapsulates the functionality for establishing and closing connections to MySQL database.

1. **RabbitMQService**

The RabbitMQService class serves to manage a channel pool with a size of 120 channels to establish a connection to RabbitmQ using the provided host information (exchange name and exchange type). It then adds each channel to the ConcurrentLinkedDeque, forming a pool. Also, this class included a close method that gracefully closes each channel within the provided pool, handling potential exceptions.

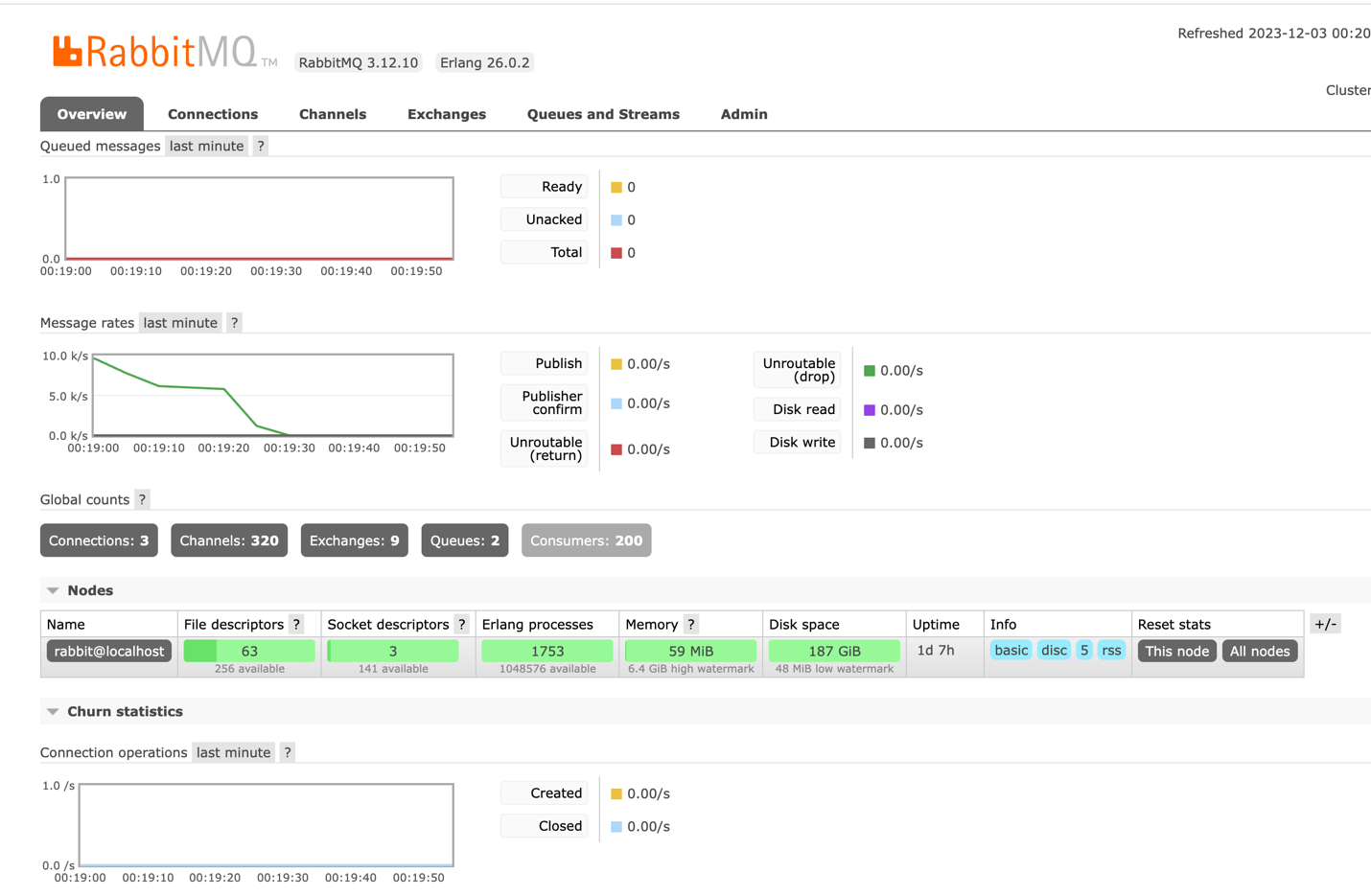
In the RabbitMQService package, the RabbitRunnable will match each message with each queue, consume and process the review message, then incrementing the number of reviews in database.

1. **Output windows for the 3 client configuration tests run against a single server/DB**

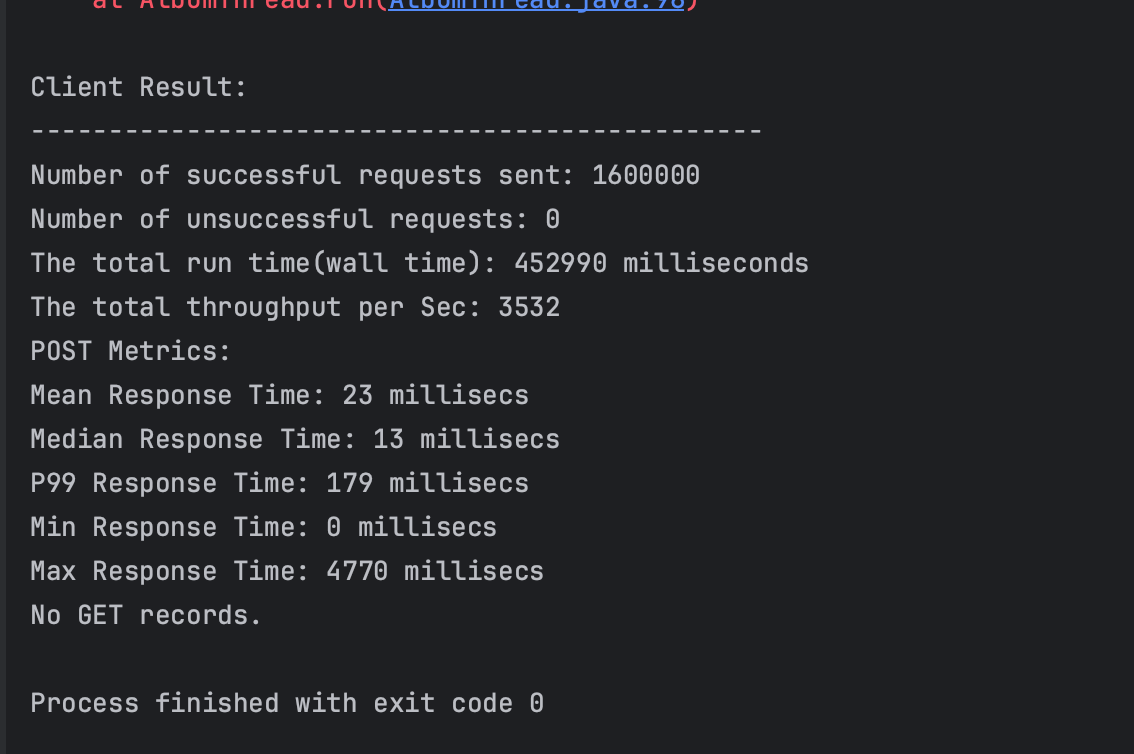
Input: 10 10 2

A screenshot of a computer

Description automatically generated



Input: 10 20 2



A screenshot of a computer

Description automatically generated

Input: 10 30 2  
