Assignment 3

1. Github: https://github.com/siqilei1004/CS6650/tree/main/assignment3

2. Server Description

The server is designed to handle the processing of posting reviews for albums. It consists of three major components: RabbitMQConsumer, ReviewDao, and ReviewServlet. These components work together to receive, process, and store user reviews efficiently.

Major classes

RabbitMQ Consumer: it is responsible for consuming messages from a RabbitMQ queue named “like_dislike_queue.” It utilizes the RabbitMQ Java client library for communication. Key attributes include:
(1)RabbitMQ Connection: The connection to RabbitMQ is established using the host, username, and password provided as environment variables.
(2)Channel Pooling: Multiple channels are created and pooled to efficiently handle concurrent message processing. The pool size is configurable.
(3)Message Processing: Messages received from the queue are split into components (likeOrDislike and albumId) and processed. The “ReviewDao” is then used to update the database with the review information.

ReviewDao: It encapsulates the data access logic for updating album reviews in the database. Notable features include:
(1)Database Connection: Utilizes Apache DBCP2 for connection pooling to enhance performance and manageability.
(2)Review Creation: The “createReviewById” method updates the album's like or dislike count based on the received information.

ReviewServlet: It is a Java servlet responsible for handling incoming HTTP POST requests related to album reviews. It interacts with RabbitMQ to publish review information for asynchronous processing.
(1)RabbitMQ Connection: Similar to the “RabbitmqConsumer”, it establishes a connection to RabbitMQ using provided credentials.
(2)Channel Pooling: Utilizes a “GenericObjectPool” to manage RabbitMQ channels efficiently.
(3)HTTP POST Handling: Receives HTTP POST requests, validates the URL path, and publishes review information to the RabbitMQ queue.
(4)Asynchronous Processing: The “publishToQueue” method sends review information to the RabbitMQ queue for asynchronous processing, ensuring the server remains responsive.

Relationships

The “RabbitmqConsumer” and “ReviewServlet” classes share the RabbitMQ connection details, ensuring seamless communication with the message broker. The “RabbitmqConsumer” relies on the “ReviewDao” to update the database based on received review information.

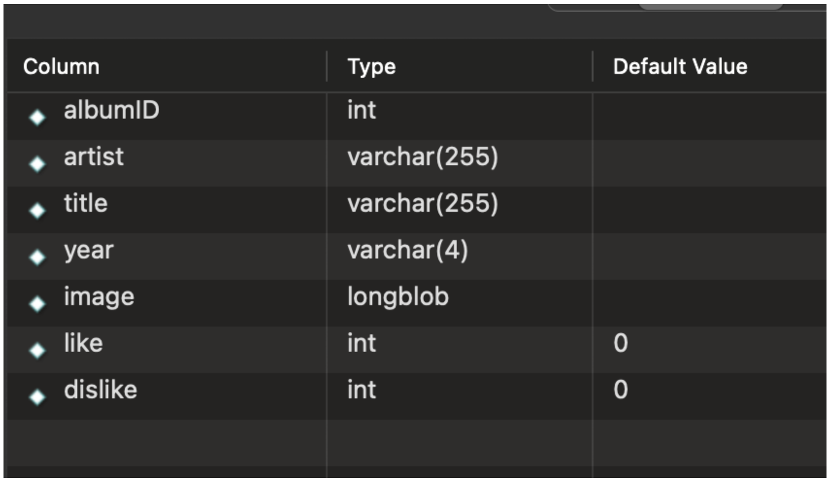
Message Flow

(1) Review Submission: A user submits a review through an HTTP POST request to the “ReviewServlet”. then the servlet validates the request, extracts review details, and publishes them to the "like_dislike_queue" RabbitMQ queue.

(2) Asynchronous Processing: The “RabbitmqConsumer” listens to the RabbitMQ queue and processes incoming messages concurrently using a pool of channels.

(3) Database Update: The “ReviewDao” class ensures database updates are handled efficiently using connection pooling and prepared statements.

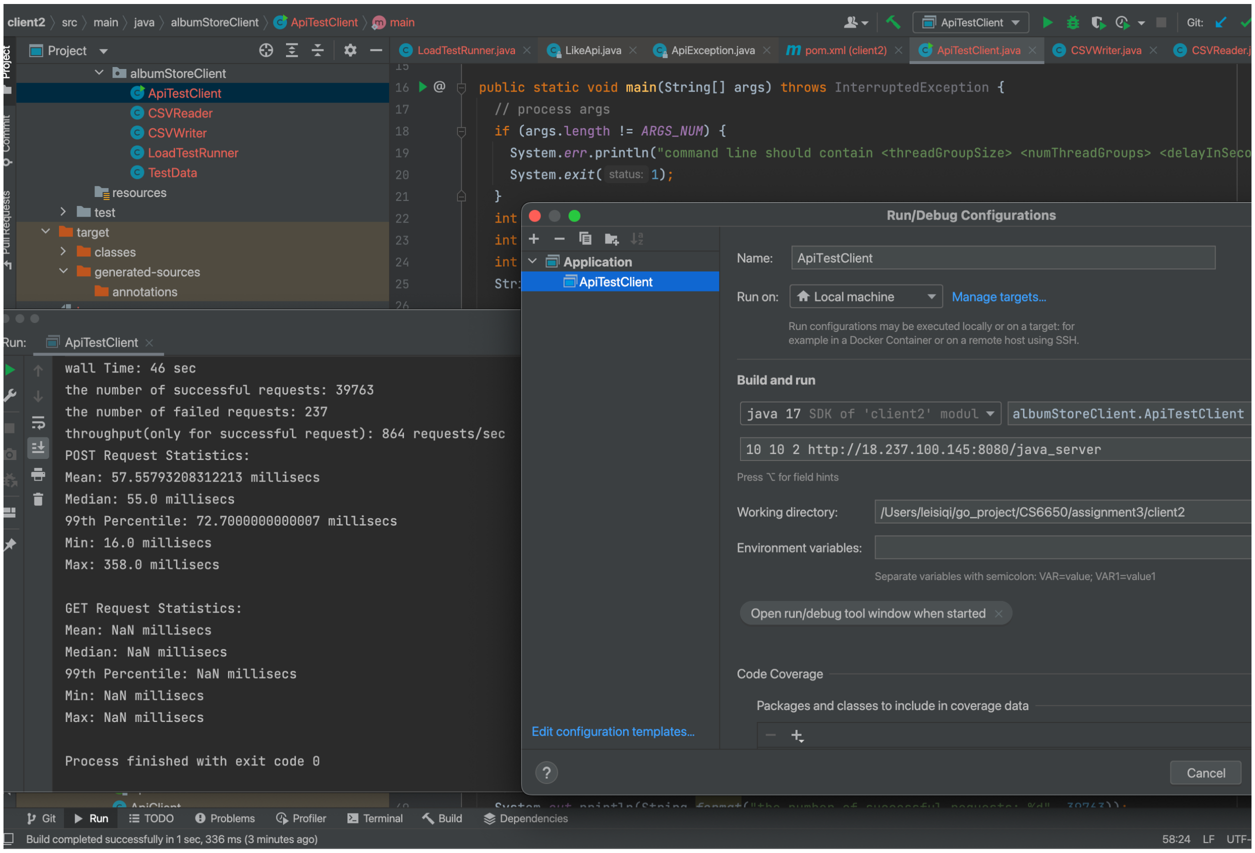
Database



Column	Type	Default Value
albumID	int	
artist	varchar(255)	
title	varchar(255)	
year	varchar(4)	
image	longblob	
like	int	0
dislike	int	0

3. Outputs

(1) threadGroupSize = 10, numThreadGroups = 10, delay = 2



```
public static void main(String[] args) throws InterruptedException {
  // process args
  if (args.length != ARGS_NUM) {
    System.err.println("command line should contain <threadGroupSize> <numThreadGroups> <delayInSeco
    System.exit( status: 1);
  }
  int
  int
  int
  Str
```

```
wall Time: 46 sec
the number of successful requests: 39763
the number of failed requests: 237
throughput(only for successful request): 864 requests/sec
POST Request Statistics:
Mean: 57.55793208312213 millisecs
Median: 55.0 millisecs
99th Percentile: 72.7000000000007 millisecs
Min: 16.0 millisecs
Max: 358.0 millisecs

GET Request Statistics:
Mean: NaN millisecs
Median: NaN millisecs
99th Percentile: NaN millisecs
Min: NaN millisecs
Max: NaN millisecs

Process finished with exit code 0
```

Run/Debug Configurations

Application
ApiTestClient

Name: ApiTestClient

Run on: Local machine Manage targets...

Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Build and run

java 17 SDK of 'client2' modul | albumStoreClient.ApiTestClient

10 10 2 http://18.237.100.145:8080/java_server

Press ⌥ for field hints

Working directory: /Users/leisiqi/go_project/CS6650/assignment3/client2

Environment variables:

Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

Code Coverage

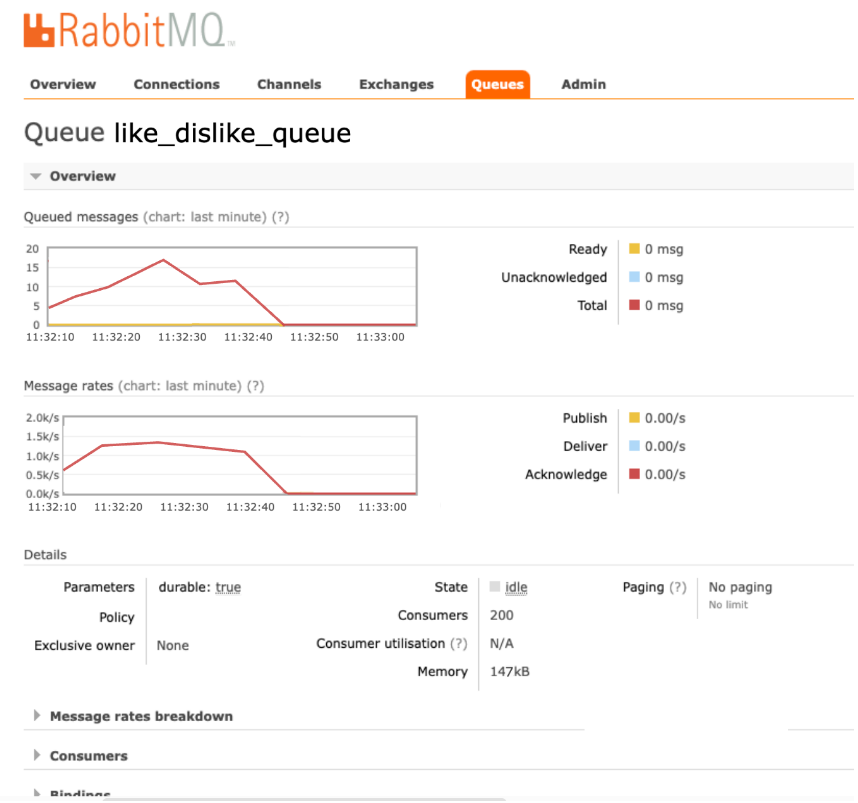
Packages and classes to include in coverage data

Edit configuration templates...

Cancel

Build completed successfully in 1 sec, 336 ms (3 minutes ago)

58:24 LF UTF-



Queue like_dislike_queue

Overview

Queued messages (chart: last minute) (?)

Ready	🟨 0 msg
Unacknowledged	🟦 0 msg
Total	🟥 0 msg

Message rates (chart: last minute) (?)

Publish	🟨 0.00/s
Deliver	🟦 0.00/s
Acknowledge	🟥 0.00/s

Details

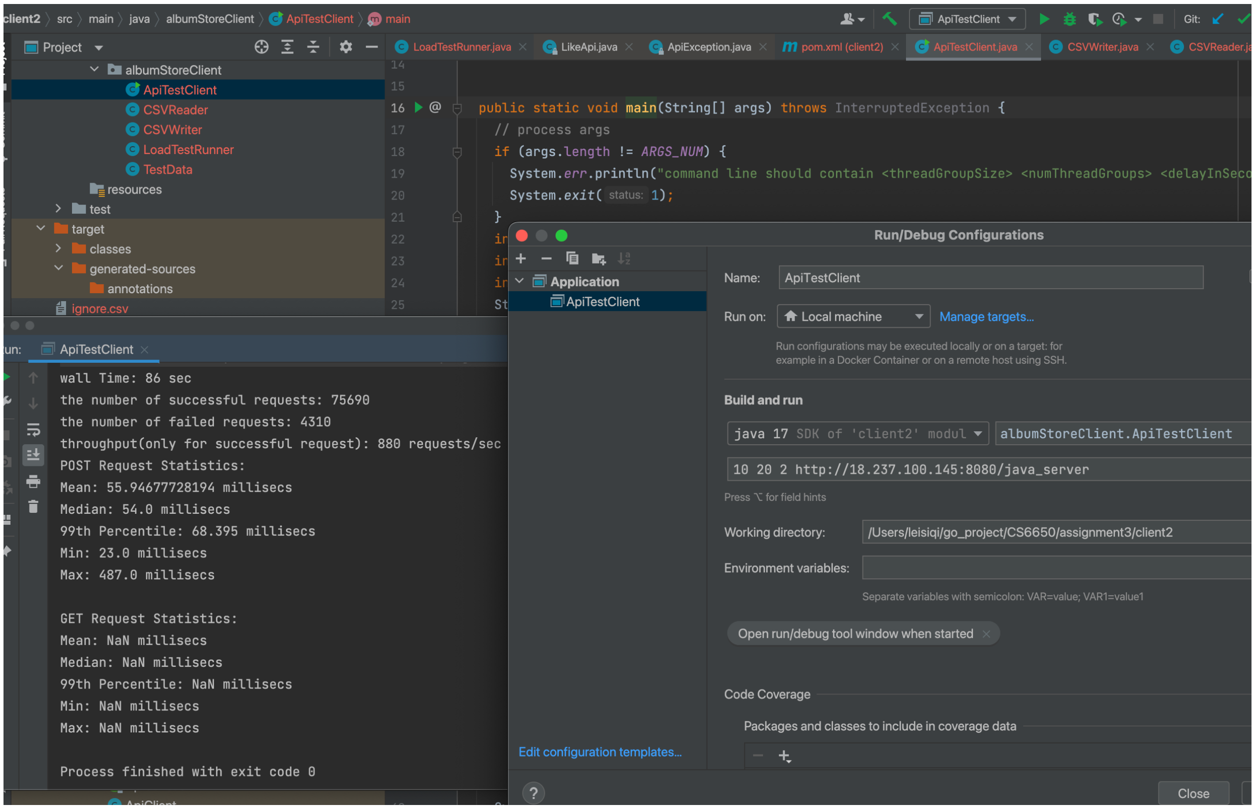
Parameters	durable: true	State	idle	Paging (?)	No paging No limit
Policy		Consumers	200		
Exclusive owner	None	Consumer utilisation (?)	N/A		
		Memory	147kB		

Message rates breakdown

Consumers

Bindings

(2) threadGroupSize = 10, numThreadGroups = 20, delay = 2



```
public static void main(String[] args) throws InterruptedException {
  // process args
  if (args.length != ARGS_NUM) {
    System.err.println("command line should contain <threadGroupSize> <numThreadGroups> <delayInSeco
    System.exit( status: 1);
  }
  ir
  ir
  ir
  St
```

```
wall Time: 86 sec
the number of successful requests: 75690
the number of failed requests: 4310
throughput(only for successful request): 880 requests/sec
POST Request Statistics:
Mean: 55.94677728194 millisecs
Median: 54.0 millisecs
99th Percentile: 68.395 millisecs
Min: 23.0 millisecs
Max: 487.0 millisecs

GET Request Statistics:
Mean: NaN millisecs
Median: NaN millisecs
99th Percentile: NaN millisecs
Min: NaN millisecs
Max: NaN millisecs

Process finished with exit code 0
```

Name: ApiTestClient

Run on: Local machine Manage targets...

Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Build and run

java 17 SDK of 'client2' modul albumStoreClient.ApiTestClient

10 20 2 http://18.237.100.145:8080/java_server

Press ⌥ for field hints

Working directory: /Users/leisiqi/go_project/CS6650/assignment3/client2

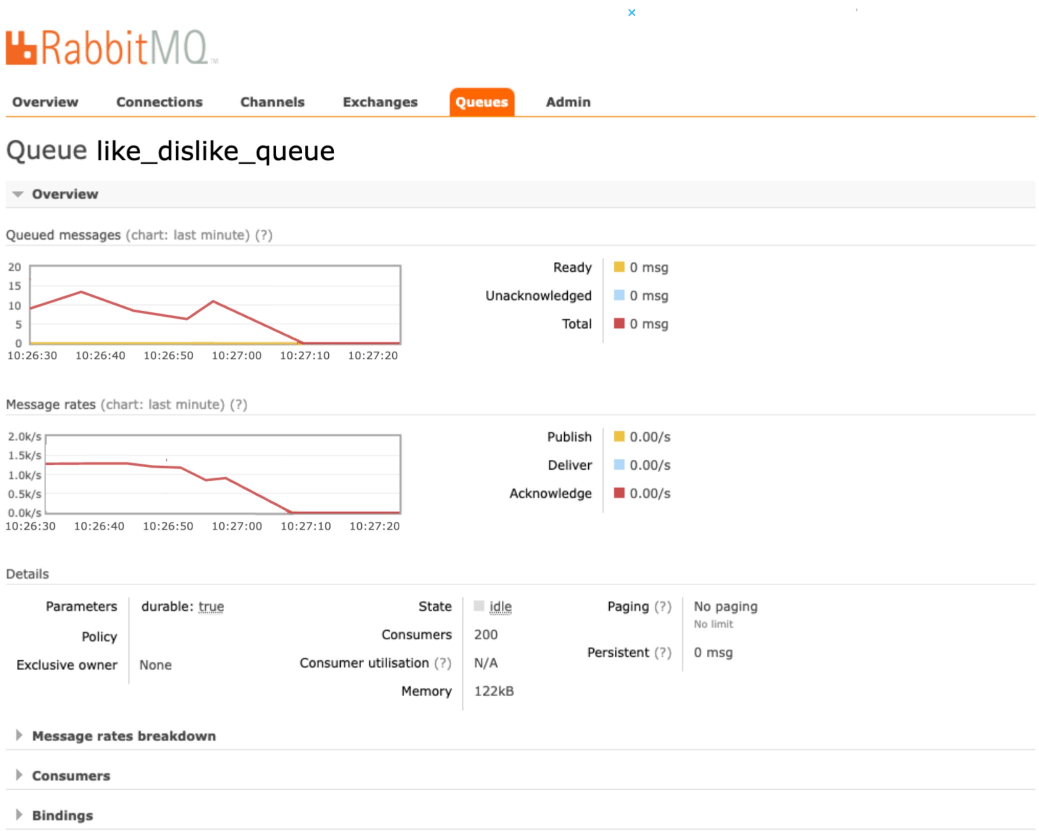
Environment variables:

Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

Code Coverage

Packages and classes to include in coverage data



Queue like_dislike_queue

Overview

Queued messages (chart: last minute) (?)

Ready	🟨 0 msg
Unacknowledged	🟦 0 msg
Total	🟥 0 msg

Message rates (chart: last minute) (?)

Publish	🟨 0.00/s
Deliver	🟦 0.00/s
Acknowledge	🟥 0.00/s

Details

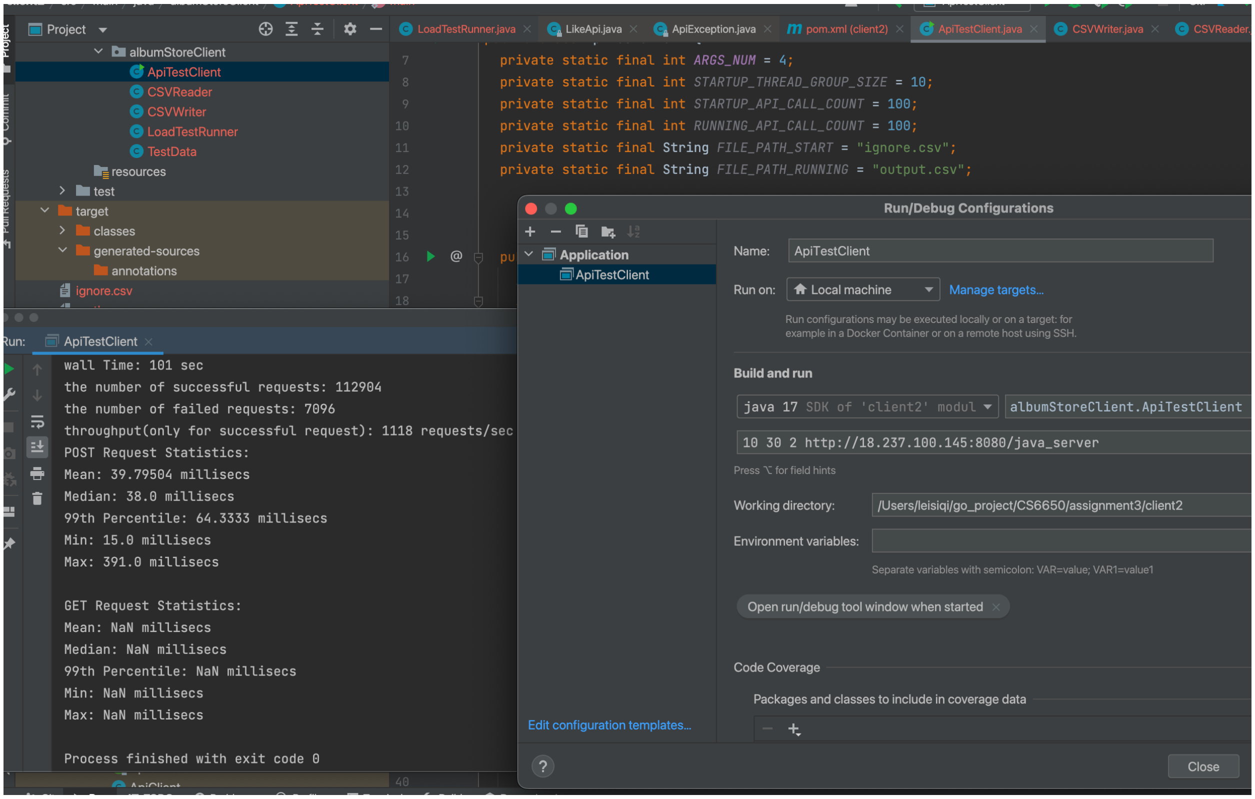
Parameters	durable: true	State	⬜ idle	Paging (?)	No paging No limit
Policy		Consumers	200		
Exclusive owner	None	Consumer utilisation (?)	N/A	Persistent (?)	0 msg
		Memory	122kB		

Message rates breakdown

Consumers

Bindings

(3) threadGroupSize = 10, numThreadGroups = 30, delay = 2



```
private static final int ARGS_NUM = 4;
private static final int STARTUP_THREAD_GROUP_SIZE = 10;
private static final int STARTUP_API_CALL_COUNT = 100;
private static final int RUNNING_API_CALL_COUNT = 100;
private static final String FILE_PATH_START = "ignore.csv";
private static final String FILE_PATH_RUNNING = "output.csv";
```

```
wall Time: 101 sec
the number of successful requests: 112904
the number of failed requests: 7096
throughput(only for successful request): 1118 requests/sec
POST Request Statistics:
Mean: 39.79504 millisecs
Median: 38.0 millisecs
99th Percentile: 64.3333 millisecs
Min: 15.0 millisecs
Max: 391.0 millisecs

GET Request Statistics:
Mean: NaN millisecs
Median: NaN millisecs
99th Percentile: NaN millisecs
Min: NaN millisecs
Max: NaN millisecs

Process finished with exit code 0
```

Name: ApiTestClient

Run on: Local machine

Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Build and run

java 17 SDK of 'client2' modul | albumStoreClient.ApiTestClient

10 30 2 http://18.237.100.145:8080/java_server

Press ⌥ for field hints

Working directory: /Users/leisiqi/go_project/CS6650/assignment3/client2

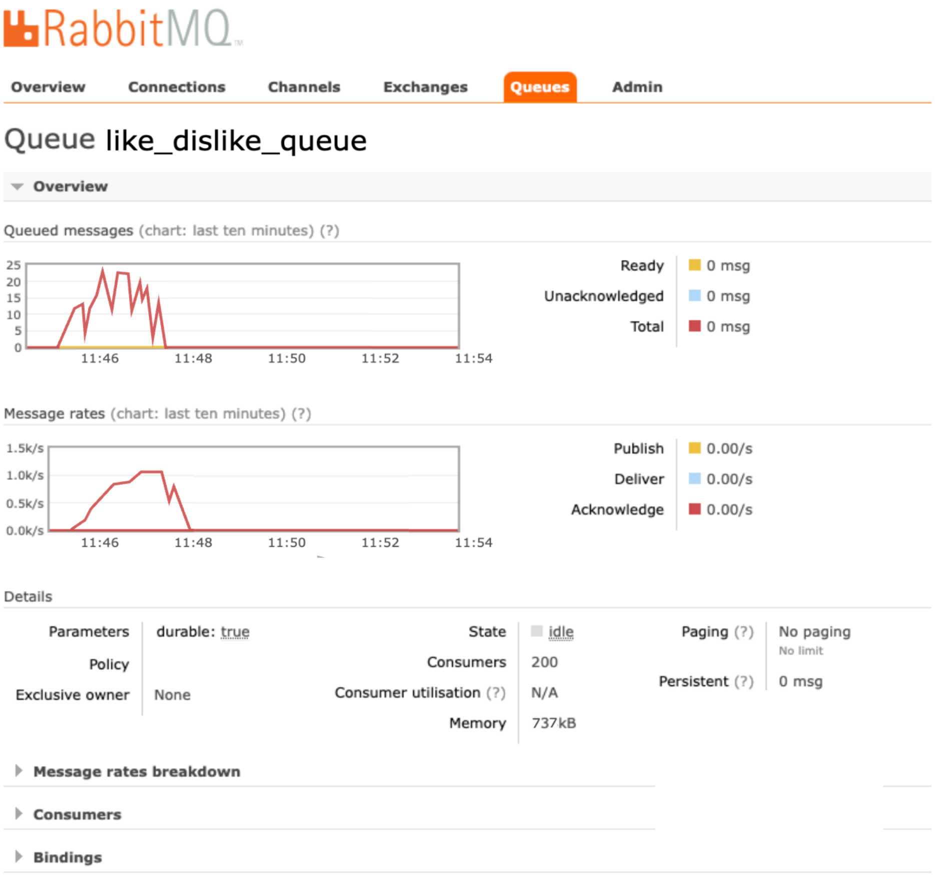
Environment variables:

Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

Code Coverage

Packages and classes to include in coverage data



Queue like_dislike_queue

Overview

Queued messages (chart: last ten minutes) (?)

Ready	🟨 0 msg
Unacknowledged	🟦 0 msg
Total	🟥 0 msg

Message rates (chart: last ten minutes) (?)

Publish	🟨 0.00/s
Deliver	🟦 0.00/s
Acknowledge	🟥 0.00/s

Details

Parameters	durable: true	State	⬜ idle	Paging (?)	No paging No limit
Policy		Consumers	200	Persistent (?)	0 msg
Exclusive owner	None	Consumer utilisation (?)	N/A		
		Memory	737kB		