Distributed Systems

Assignment 3

Divricean Dan Alexandru

Group 30444

Table of Contents

[1. Requirements 2](#_Toc468645026)

[2. Conceptual architecture of the distributed system 2](#_Toc468645027)

[3. UML Deployment diagram 3](#_Toc468645028)

[4. Build and execution 3](#_Toc468645029)

# Requirements

Design, implement and test a distributed system that uses MOM to create an asynchronous communication between the client ( message producer ) and the server ( message consumer ).

# Conceptual architecture of the distributed system

The architecture on which the system is based is the client – server architecture. The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

The communication between the client and the server is done by using MOM. MOM provides software elements that reside in all communicating components of a client/server architecture and typically support asynchronous calls between the client and server applications. MOM reduces the involvement of application developers with the complexity of the master-slave nature of the client/server mechanism. The software used for assuring the communication between the client and the server in RabbitMQ. RabbitMQ is open source message broker software ( message-oriented middleware - MOM ) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.

The application consists of two parts: the client and the server. As mentioned above, the communication between the two parts is made by using RabbitMQ.

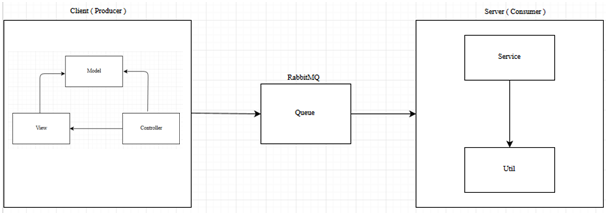
The server is a simple application, that has two functionalities:

* It sends and e-mail to all the subscribers, when a new DVD is added
* It creates a text file with the details of a DVD, when a new DVD is added

The server is based on the Layers design patterns. It has two layers: the service layer and the util layer. The sevice layer uses functionalities provided by the util layer.

The client is a web application. The MVC design pattern was used. This pattern separates the application into three parts:

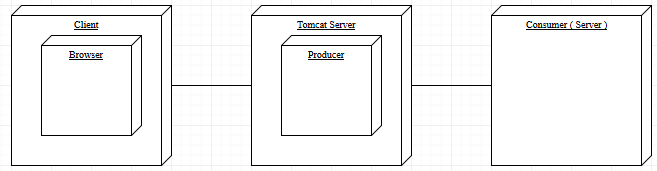
* The model - Model represents and object carrying data. It can also contain logic.
* The view – View represents the visualization of the data.
* The controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.



# UML Deployment diagram

The application consists of two modules: the consumer ( server ) and the producer ( client ). These modules can be run of different systems.

In the picture presented below, you can see the deployment diagram of the application.



# Build and execution

In order to build the source code, you should have installed on your computer the following:

* Eclipse IDE
* Apache Tomcat 8.0
* RabbitMQ

In order to run the application, having the source code, you should follow these steps:

* Start RabbitMQ
* Open Eclipse IDE
* Import the project into Eclipse as a Maven project
* Right click on the Producer module and select Run as... -> Run on server
* Select the Apache Tomcat server
* Right click on the Consumer module and select Run as... -> Java Application

If the application is deployed on the server, it can be accesed by opening the browser and typing in the following URL: <http://188.24.8.54:8080/Assignment3.2Producer/>.

In both the cases, the following page should open:

