DISTRIBUTED SYSTEMS

**A3.2: Asynchronous Distributed System Application**

**using Java or .Net Messaging Frameworks**

Acu Raul-Mihai

30442

**Table of Contents**

1. **Project requirements**
2. **Implementation details**
3. **Conceptual architecture**
4. **Deployment diagram**
5. **Project requirements**

**Functional requirements:**

 The application is used by a DVD store administrator

 The administrator must send notification to its customers when a new DVD is available

 The information about the new DVD must be saved in a text file

 Each time new information about a DVD is introduced in the system, the application must

send automatically notification e-mails to all the subscriber customers to notify them about

the new item

 Each time new information about a DVD is introduced in the system the application must

create automatically a text file and write the information about the DVD in it **Implementation technologies:**

Use one of the following technologies:

o For message producer and consumer: Java or .NET

o For message queue:

 Java: JMS, Java API of RabbitMQ

 .NET: MSMQ, .NET API of RabbitMQ

1. **Implementation details**

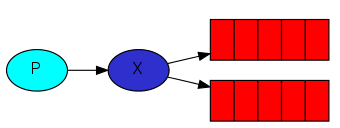
**RabbitMQ:** RabbitMQ is lightweight and easy to deploy on premises and in the cloud. It supports multiple messaging protocols. RabbitMQ can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements. A producer is a user application that sends messages.

A queue is a buffer that stores messages.

A consumer is a user application that receives messages.

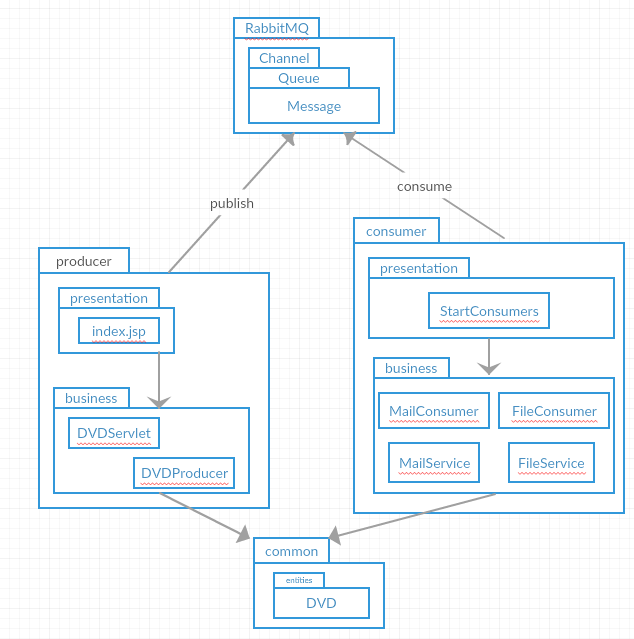
The core idea in the messaging model in RabbitMQ is that the producer never sends any messages directly to a queue. Actually, quite often the producer doesn't even know if a message will be delivered to any queue at all.

Instead, the producer can only send messages to an exchange. An exchange is a very simple thing. On one side it receives messages from producers and the other side it pushes them to queues. The exchange must know exactly what to do with a message it receives. Should it be appended to a particular queue? Should it be appended to many queues? Or should it get discarded. The rules for that are defined by the exchange type.



1. **Conceptual architecture**

Conceptual architecture is a form of architecture that utilizes conceptualism, characterized by an introduction of ideas or concepts from outside of architecture often as a means of expanding the discipline of architecture. This produces an essentially different kind of building than one produced by the widely held 'architect as a master-builder' model, in which craft and construction are the guiding principles



1. **Deployment diagram**

