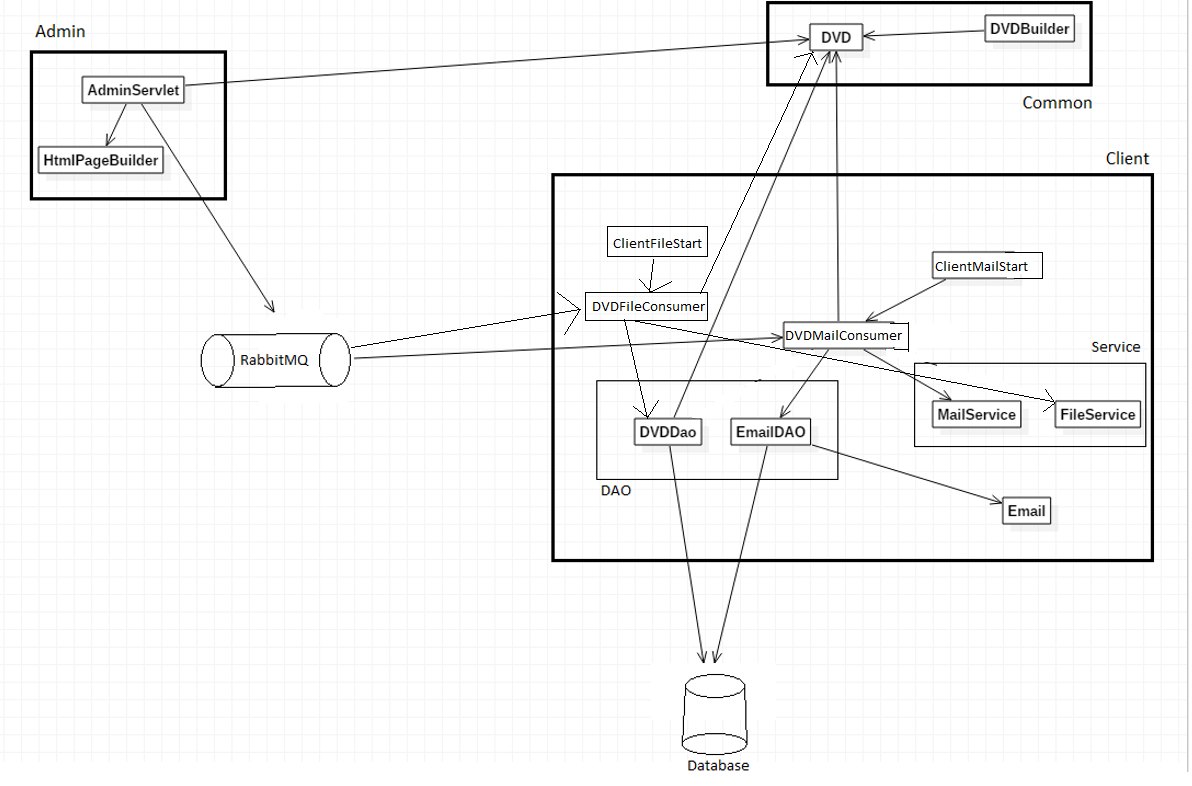
DISTRIBUTED SYSTEMS

Assignment 3

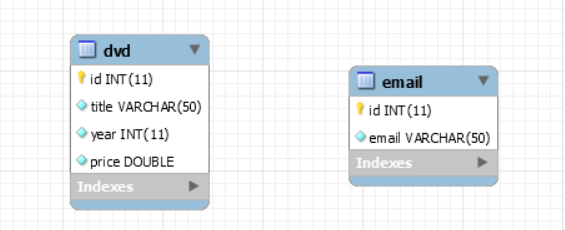
Asynchronous Communication using Messaging

A3.2: Asynchronous Distributed System Application using RabbitMQ

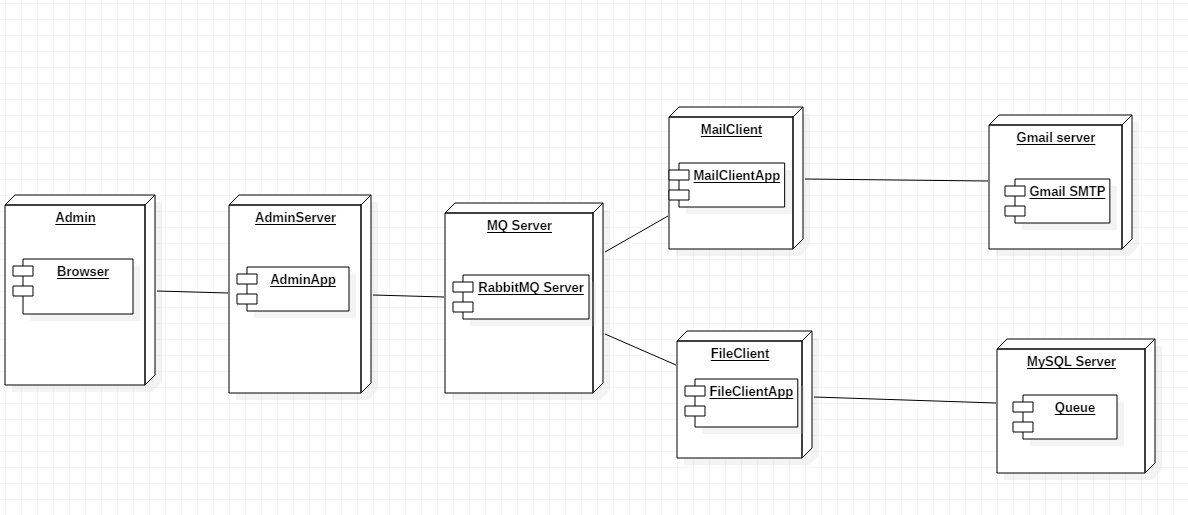
* Conceptual architecture of the distributed system:

The distributed system is conceptually split in three logical components which are further complemented by two other “outside” components. The first logical component is Common, which contains the classes which are used in multiple parts of the applications (on multiple machines). In the case of this application it is only the DVD entity class and its builder. The Admin component contains the classes required for the web-based application, namely the AdminServlet servlet and a Html page builder. Besides handling Http requests, the servlet also sends the messages to the RabbitMQ queue. The Client component contains the classes that interact with the database and handle the messages received from the queue (send mails and create files).

* Database diagram:

The database diagram is fairly simple, it only contains two tables: one for storing the subscriber email addresses and one for storing the DVDs received from the message queue (for a potential future use). There are no relationships between tables.

* Deployment diagram:

The deployment diagram is pretty large as we can potentially have 6 separate machines running parts of our distributed system. The administrator uses a browser from their computer to access the adminapp web application which resides on yet another computer. The application then sends a message to a remote message queue. The message is then retrieved by the MailClientApp and by the FileClientApp. The first app sends email with the data, while the second creates locally a file with the data and stores it in a database (which could be on another machine). In order to send emails, it must access the Gmail SMTP server. In the current configuration, everything (besides the SMTP server) runs on the same machine, however this can easily be changed with minimal effort.

* Build and execution considerations: Before running the project, make sure the RabbitMQ server is installed on your machine
  + Opening the project:
    - Open IntelliJ IDEA
    - File | Open
    - Browse and select the directory containing the project
    - Click “Open”
    - Go to pom.xml file, double click to open it and right click on the editor
    - Select Maven | Reimport to make sure that all the required libraries are included
    - Open the hibernate.cfg.xml file from src/main/resources from the clientapp module
    - Modify the appropriate fields with the data required to access the database schema
  + Running the project:
    - Navigate to the “ClientStart” class from the clientapp module
    - Run | Run “ClientStart”
    - Navigate to the “AdminServlet” class from the adminapp module
    - Go to Run | Edit Configurations
    - Click on the Green plus (+)
    - From the dropdown list, select Tomcat Server | Local
    - (Optional) Assign a name to the run configuration
    - Change the field from Open browser to “http://localhost:8080/admin”
    - Click on the “Fix” button from bottom right
    - Select (Project name):war
    - Click “Ok”
    - Click Run | Run (Configuration name) – first option
    - A browser page should automatically open with the default URL (<http://localhost:8080/>admin is default). If not, open a browser and go to the aforementioned URL.