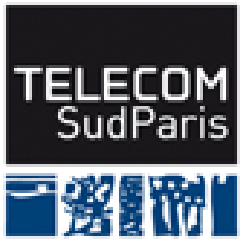

Middleware for Distributed Applications
Microproject Report

KAZMI, Syed M. M.

MACHADO, Rebeca

Contents

Microproject Report

1. Introduction and project goal

The goal of the project is to develop a mailbox application using Middleware technologies (RMI and Web Services such as RESTful and SOAP). The mailbox application consists of Mail Box Manager, directory manage, an Administrator client and clients. Mail Box Manager handles user mailboxes and a common newsgroup, which may be read by every user. A directory manages the users, which may access the mailbox and their rights to access newsgroup. Only some users have the right to send messages to the common newsgroup.

2. Software architecture

2.1 General architecture

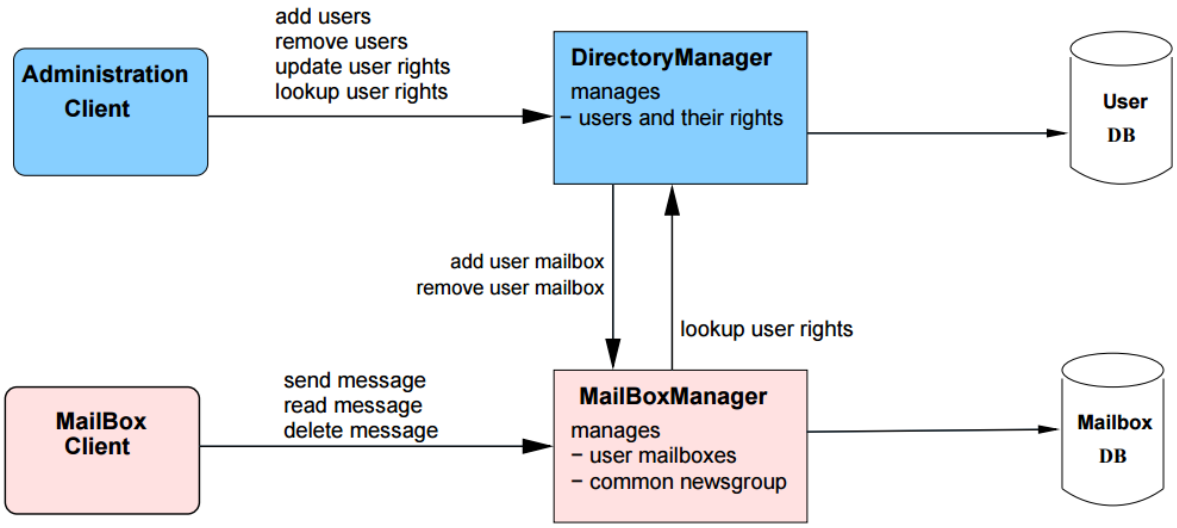

Figure 1: Proposed architecture of the system in the project formulation

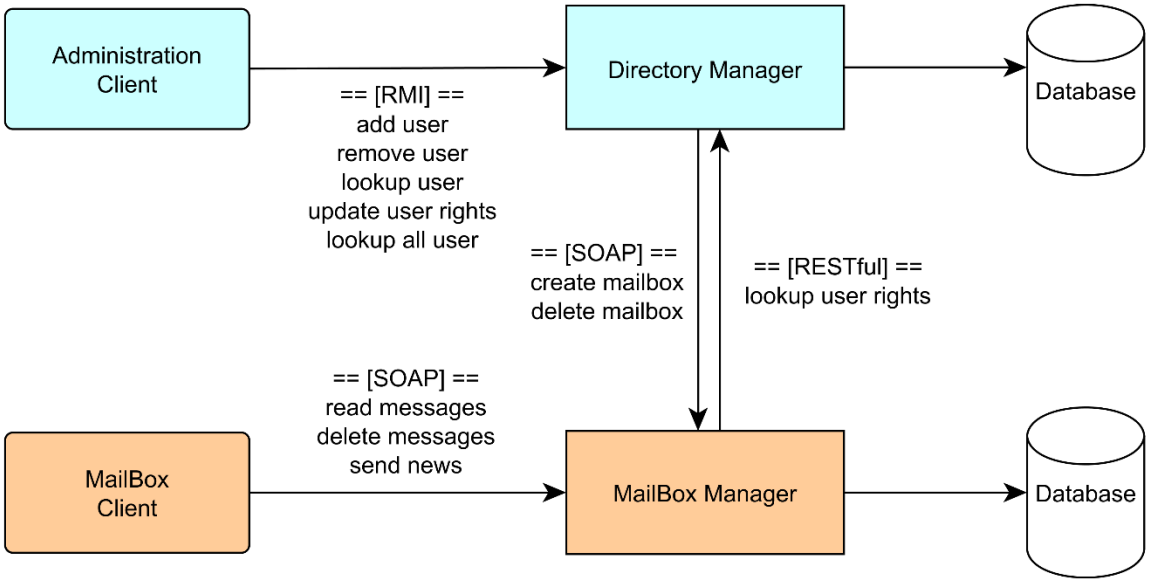

Figure 2: Actual architecture of the system

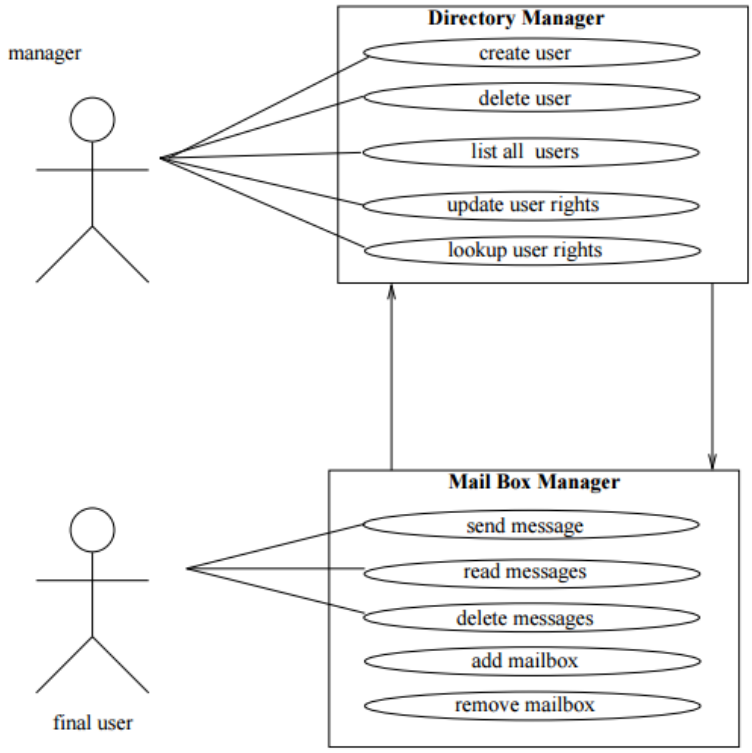

Figure 3: General Use Case diagram

2.2 User manager architecture

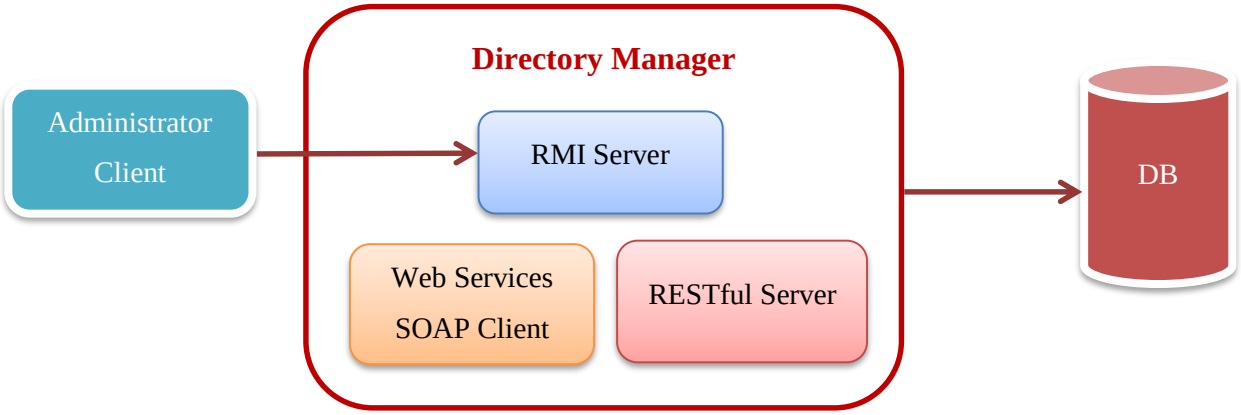

Figure 4: Directory Manager schematic diagram

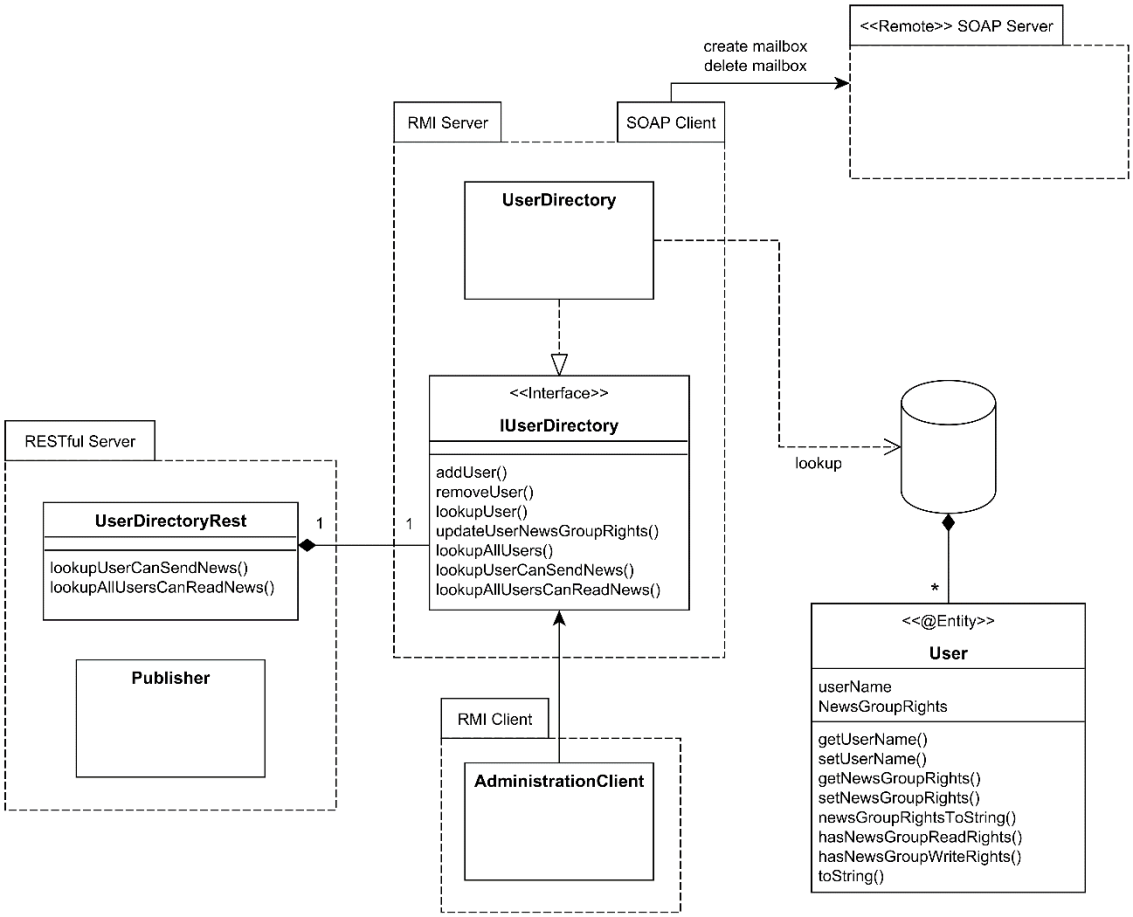

Figure 5: Class diagram of Directory Manager

User Directory manages addition and removal of users and their rights for common news group box and keeps that information in a persistent storage (database). An administration client uses all these provided operations. The Directory Manager also publishes methods for users' rights lookup by MailBox Manager.

User Directory Manager hosts two servers: RMI server and RESTful server while it acts as client for SOAP Web Service (held by the MailBox Manager). Directory Manager is deployed on Glassfish and the Administrator Client access it using RMI (JavaEE). It hosts RESTful services for user rights lookup for read/write emails on the common newsgroup. Directory Manager calls creation/deletion of users' mailboxes by acting as SOAP client.

2.3 Mailbox manager architecture

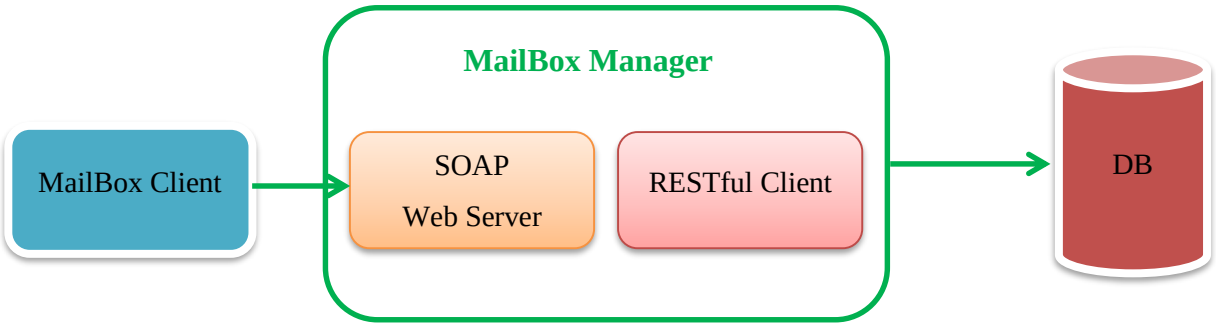

Figure 6: MailBox Manager schematic diagram

MailBox manager is the key part of the mailbox application. It handles a mailbox for each client and provides methods to send, read and delete users' emails. It also has a common newsgroup for broadcast messages, which may be read by all users, who have common newsgroup read rights in User Directory Manager while only those clients who have writing access can send messages to the common newsgroup.

A MailBox Client communicates with the MailBox Manager by SOAP Web Services, allowing the sending, reading and deleting of messages. The MailBox Manager also acts as a RESTful client to obtain a user's rights to access the news group from the Directory Manager.

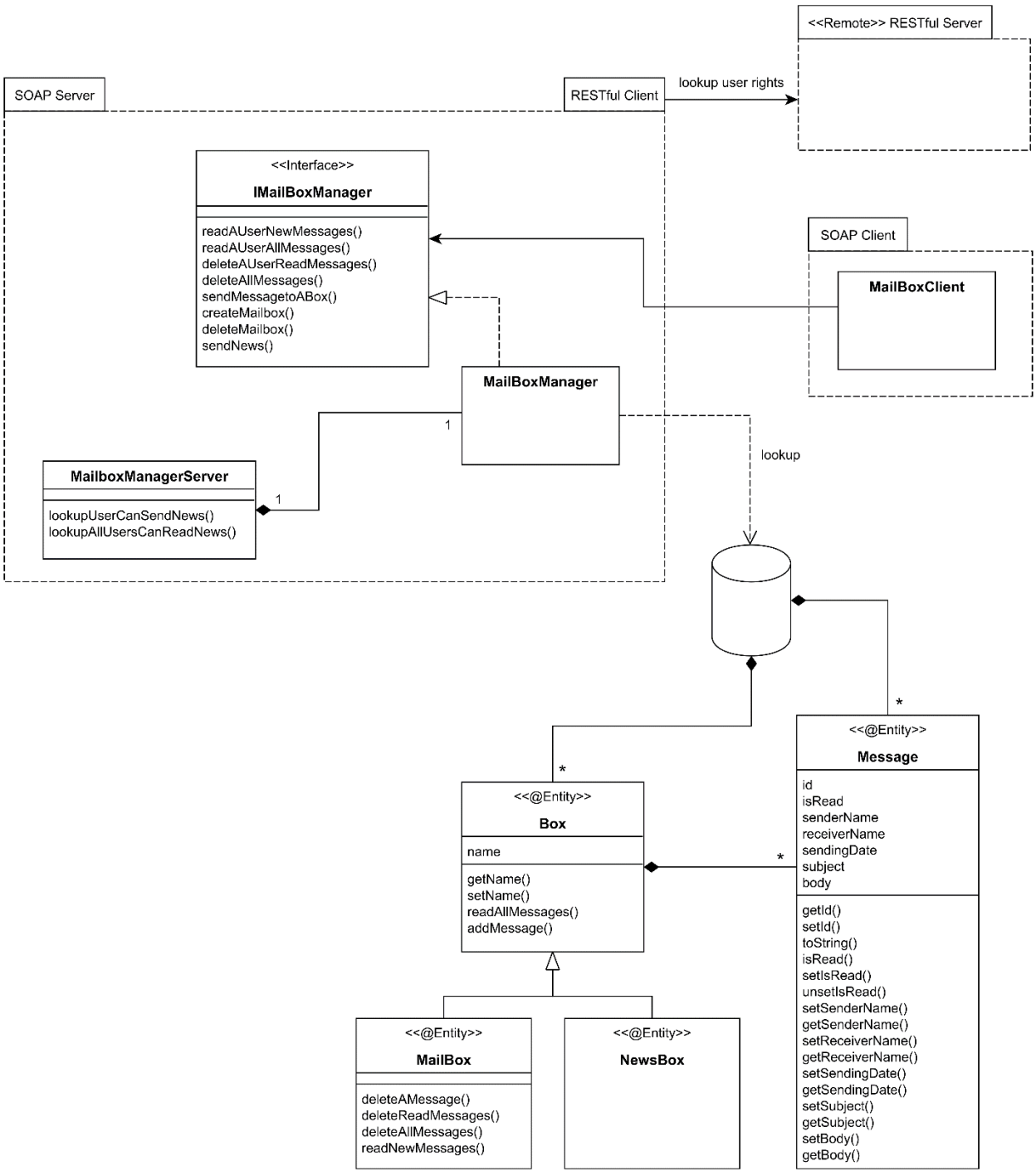

Figure 7: Class diagram of MailBox Manager

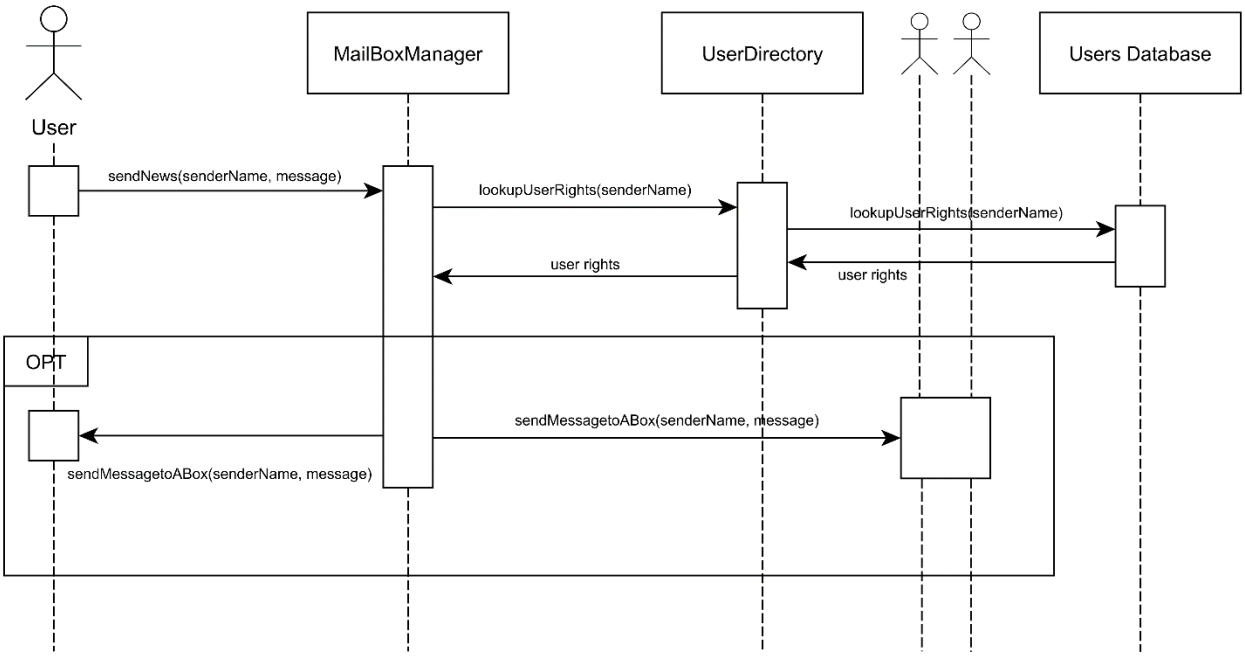

Figure 8: Sending a message to the news group

3. Technical decisions

Following the formulation of the project, we took in consideration JavaEE and Web Services. As for what we understood, the Administration Client should connect through RMI to the Directory Manager and the MailBox Client through Web Services to the MailBox Manager. For this last one, we had the option of picking SOAP or RESTful, but since we were asked to implement RESTful in one of our project's component, we assumed the Web Services connection in the MailBox component referred to SOAP. Hence, there was only one connection decision missing between the MailBox Manager and the Directory Manager; for which we chose REST. We also considered some other aspects explained hereunder to implement it this way:

1) SOAP: We use the SOAP Web Services at MailBox manager because it could also provide authentication mechanisms if someone extends the application so the clients could access their mailbox only by logging in.
2) RESTful: REST is faster than SOAP, useful since our application is running in a distributed environment and has to look up for user rights every time they want to send broadcast messages.

We also decided to implement the News Group as a distribution list. When a user sends a message, as you can see in Figure 8, if it has the rights to send news, the messages will be broadcasted to all the users in the system with read access rights. Then they can read the news from their mailboxes. This is similar to the subscriptions in DEBS learnt during classes.

4. Difficulties

1) Integrating SOAP Web Services with a JDBC is quite hard. Unfortunately, we had to specify the login information to the database in the persistence file for it to work correctly.
2) To implement clients and servers using different middleware technologies in the same class/package was also difficult. We faced a lot of compiling and running issues, especially with the .jar dependencies.
3) The concurrent modification of the database was also a problem. We had to implement a structure provided by java that allows to modify an array at the same time it is read with synchronous methods.

5. Design patterns

As for what we did in both major components (managers) we used a Façade pattern to correctly publish the methods used by their clients. The clients of the Directory Manager get the IUserDirectory interface to use its methods. The SOAP Web Service implemented for the MailBox Manager works differently, as it needs to publish every method of the manager for the clients to get them.

6. User manual

6.1 Automated way

Inside the root of the project, there are two files called “demo.sh” and “hosts”. This last one specify a list of 4 hosts to distribute the application on following this order:

- Directory Manager address/hostname
- Administration Client address/hostname
- MailBox Manager address/hostname
- Mailbox Client address/hostname

They will be executed remotely by ssh. To run the project, it will be as simple as type

>$ *sh demo.sh . hosts*

NOTE: The point specifies the path of the Microproject. All three folders should be in the same directory.

6.2 Manual way

The components should be run in this order in order to make the application work correctly:

1) Choose the addresses and hostnames of the 4 components. They could be all different or the same. Each component will be run at localhost, therefore a ssh should be done first to distribute it. Write down these addresses.
   NOTE: Do not pass "localhost" as arguments. Rather pass $HOSTNAME.

2) *cd* to the MailBoxManager directory and type
   >$ *sh run_server.sh <address of the directoryManager>*

3) *cd* to the DirectoryManager directory and type
   >$ *sh run_server.sh <address of the mailBoxManager>*
   This will also run "asadmin start-domain domain1".

4) *cd* to the DirectoryManager directory and type
   >$ *sh run_client.sh <address of the directoryManager>*

5) *cd* to the MailBoxClient directory and type
   >$ *sh run_client.sh <address of the mailBoxManager>*

For more information about the project and the contents, please see the code and the README.txt provided with it.