Master of Science - CCN2

Middleware for Distributed Applications

CSC 7321

MICRO PROJECT

REPORT

Submitted by:

- Abdallah Sobehy
- Mostafa Fateen

Submitted to:

- Prof. Chantal Taconet
- Prof. Sophie Chabridon

Date: 12-Nov-15

Contents

System Architecture

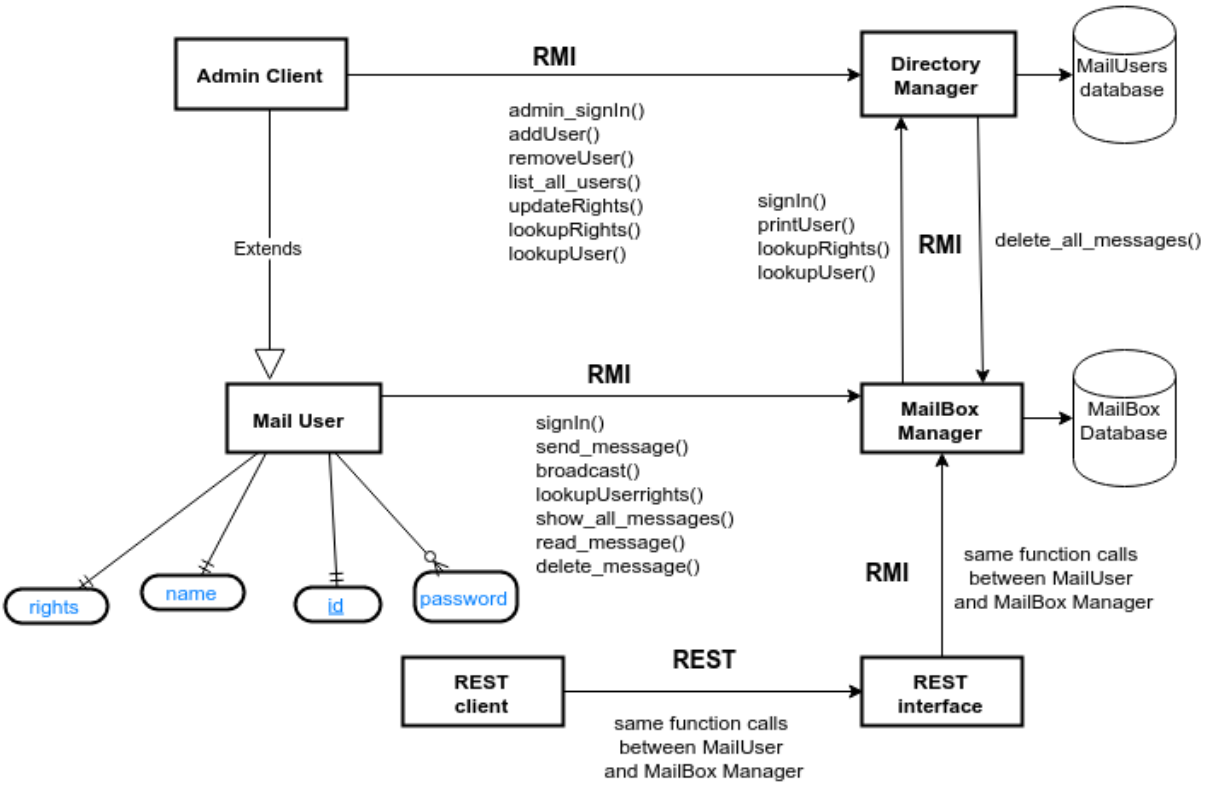

Figure 1

Following the diagrams presented in the project description [1] and adding some features according to our design of the system we designed the Mail Box application as shown in figure 1.

System Components

a) Mail User: a class that contains information about mail users e.g. name, password (which are used for sign in), id, rights [admin, broad cast user, normal user].

b) Admin: Inherits from Mail User and has functions only available for the admin user e.g. Add user, remove user, update a user's right …

c) Directory Manager: Stateful EJB for persisting users in database. Accessed by the admin to create, remove, update users. Also, Mailbox manager accesses it to query information about users e.g. rights, checks for username and password for sign in....

d) Mailbox Manger: Stateful EJB for persisting messages in the Mail Box data base; all messages are stored in the data base. Messages have Id, sender, receiver, status (read/unread).

e) REST client and interface: client and server which are used to provide a RESTful interface for the mailbox manager client (functions provided are the same as between the Mail User and Mail Box Manager).

System Behavior

- Initially, databases are empty; no users nor messages are stored. Since only the admin can add users to the system, we decided that the first user to try to sign in to the Directory manager is made an admin with the given user name and password. Afterwards, any sign in operation is checked in the database for a verification for the user.

- Admin can add users specifying user name, password, rights.

- Admin can update users' rights as well as delete users

- From the Mail User client as well as the RESTful client can sign in to the system through the Mailbox manager to read/send/delete messages.

Design Patterns

The main design pattern used was the **facade** design pattern. Which as described in [2], is a structural design pattern which hides the complexity of the system form the client through an interface.

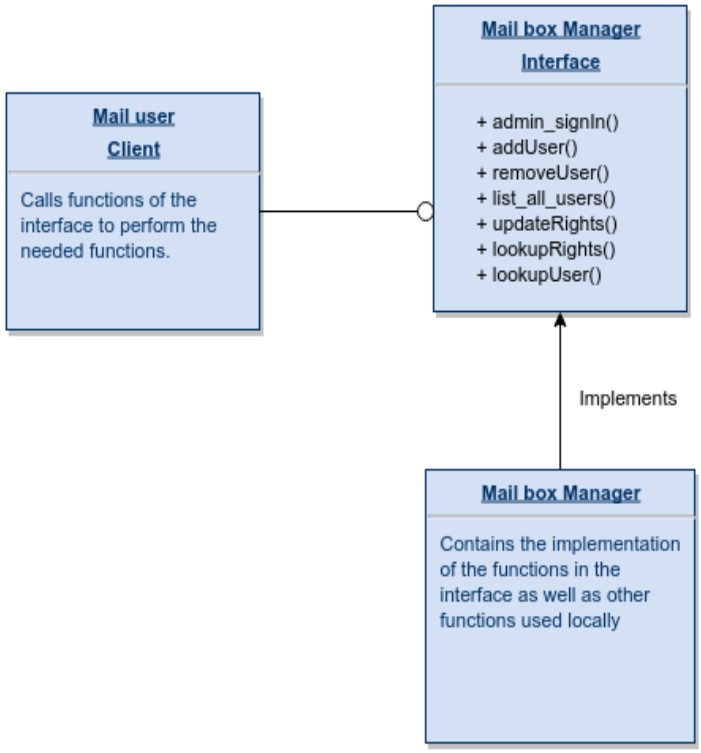

Figure 2

As can be seen from Figure 2 we provide an interface for the client which hides the complexity of the server implementation.

Baring the **separation of concerns** in our mind while designing the project we separated the implementations of entities from the clients. As a result, the Mail User client and the admin client do not need to know or import the entities' classes (MailUser, Message classes).

Technologies Used

To realize the mailbox, we used:

- Glassfish: to deploy the Mailbox. Directory managers.
- MySQL Database named: INF7321_6 to write the users in MAILUSER table and messages in Mail Box table.
- JPA & JPQL: to create table in MySQL databases
- Jersey RESTful framework & JAX-RS to provide RESTful interface for the Mailbox Client.
- Eclipse IDE & ant

Difficulties and Solutions

Difficulties	Solutions
Initializing user in an empty database	The first user that signs in becomes an admin with the chosen username and password.
Persisting MailUser (parent class) ,Admin and BcastUser (child classes) in the same table	used annotations (@Inheritance, and @DiscriminatorColumn) to specify the relation between Admin and mail user and the column (user rights) which discriminates between the two entities.
The messages of a deleted user are not removed from the database	When directory manager deletes a user it calls the Mail box manager to delete all messages where the user is the receiver.
Adding the RESTful interface to the mailbox client	The most complicated part of the project, we had to search for needed jars to be able to run it. To call a function by a URL the user has to specify login, password, function to do and input parameters

Design Decisions

- The password of the user in the interactive mode of the Admin and Mail user clients is hidden while writing it on the terminal.
- Both Mailbox and Directory managers are stateful to have a session for each signed in user.
- For simplicity all messages are stored in one table and retrieved according to the sender or receiver or id.
- In the URL all the input to a given function is given in an order and separated by ':' please check the REST server for more details.
- A user can only view the received messages or the broadcast messages. While the admin has the right to view all messages.
- A user can only delete received messages or messages he sent to the newsgroup. While the admin can delete any messages.
- The user name and password should not include ':' or ';' or '/' because they are special characters in URL format.

Added features

we provide an interactive mode for both the Admin and Mail User client.

User Manual

To run anything in the project first go to its directory and follow the following steps

1. $ asadmin start-domain domain1
2. $ asadmin start-database
3. $ ant build
2. $ ant deploy
3. then follow the instructions in the following table

What to run	How to run it
The automatic client of directory manager	$ java client_dm.AutoClient_dm
The interactive client of directory manager	$ java client_dm.Client_dm
The directory manager from a remote machine	$ sh remoterun.sh
The mailbox manager from browser	$ ant runRestful
The automatic client of mailbox manager	$ ant runRestful
	And in a new terminal run:
	$ java client_mbm.RestClient_mbm
The interactive client of mailbox manager	$ java client_mbm.Client_mbm

Notes:

1)Before running the automated client of the **directory manager** there should be at least an admin client called “Mostafa” with password “1234567890” in the database which should be already existing in the database but could be added by the interactive client if it wasn’t in the database.

2)Before running the automated client of the **mailbox manager** there should be at least:

a) An admin called “Mostafa” with password “1234567890”
b) A bcast user called “Abdallah” with password “321”
c) A normal user called “Mahmoud” with password “123”
d) A message to “Mostafa” with ID = “3851”

References

[1] S. Chabridon, C. Taconet, Micro project subject for middleware for distributed applications course: http://www-inf.telecom-sudparis.eu/COURS/CSC7321/Chap/sujetmicroprojet-chap.pdf