MICRO PROJECT

SOPHIE CHABRIDON, CHANTAL TACONET

CCN/CSC7321

Authors

Monica Cecilia ALVARADO XITUMUL
Juan Rodolfo MEJIA ROJAS

Micro Project Report

Abstract

This report aims to show the development process of a specific distributed system , we will enumerate and demonstrate the solution we developed for this project. The project involves a distributed system which will be transparent to the client and be presented as one single service. Along with a general diagram representing the system architecture of our system we will demonstrate the design of our solution listing the design patterns and technologies as well as the difficulties we fronted during the development of the project.

Introduction

The objective of this project is to demonstrate our skills acquired through the different lectures and labs from the module Middleware for Distributed Applications. On a bigger scope we need to develop a distributed system for a mail server. We need to emulate the functionalities of a mail service with access control, all regular mail functionalities including a group recipient. This group recipient is referenced through the report as the newsgroup, which some user have access to read and write. During the development of the project the most challenging part was getting all the pieces of our software together in one single service, after many attempts we were able to learn how to make each gear of our software work together which is an important goal of distributed systems.

Background

The description of the system is simple, we need to develop a system which will act as a mail manager. This mail manager will have two different services one which handles only the mail distribution and other that handles the users and their permissions. This two services will team up to be presented as one integrated service for the client to use to send and receive mails. This connection of both manager is completely transparent for the client which only has to connect to a public interface to do the regular mails stuff. Of course since we are handling mails here we need to have a persistence solution to have all mails safe in a robust data structure. The user manager is referenced in the report as the directory manager, this manager needs to handle user authentication and access control. Users can have permissions to write to a newsgroup and also have permission to receive emails from the newsgroup. This manager will communicate with the mailbox

manager to determine which users does actually get permissions to receive the messages sent to the newsgroup in his personal mailbox.

After the different managers were explained we can proceed to the two differents types of clients this distributed system has. The project required us to have a administration client that is able to create,modify and delete users and the regular mail clients which only have functions to send, receive and delete mails.

After all the project was done we can conclude that difficultest part is putting all the pieces of software together, making each piece and making it work was relatively easy but adding all the pieces of system was the most challenging part of the project. We fronted some other minor difficulties that were basically solved by tracking down the errors.

Analysis and Design

Now that the concept of the project is explained we can easily understand the system architecture diagram, this diagram presents our exact approach including the technologies used for each component of our project.

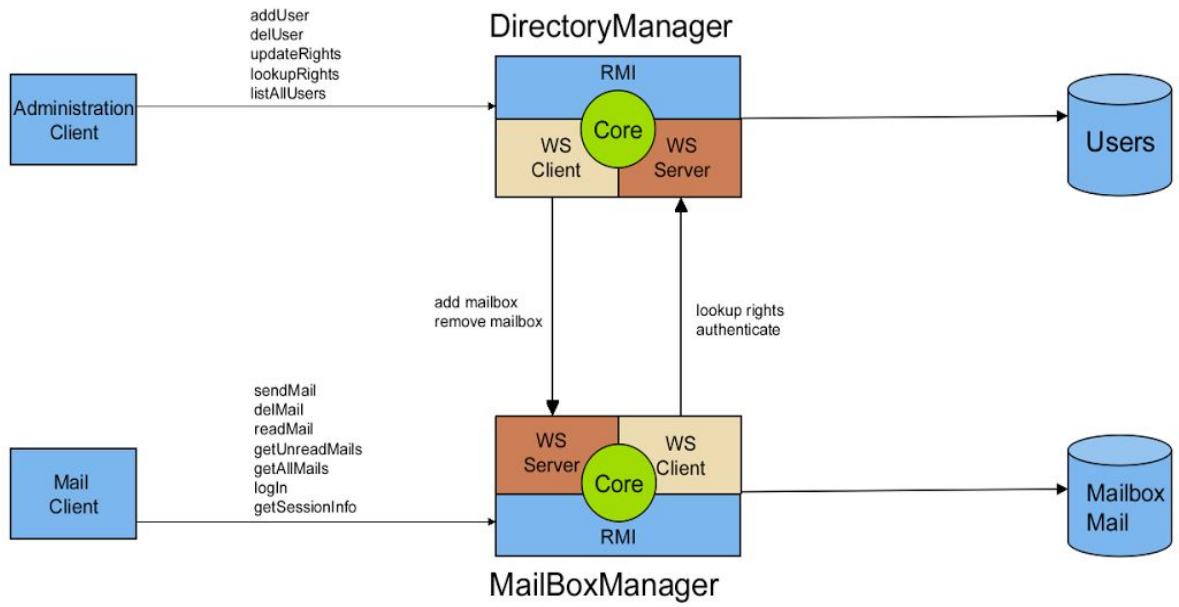

Figure 1: General architecture of the system

In Figure 1 we can see how each client uses java remote interfaces to communicate with the different managers in the system using Remote Method Invocation (RMI). The “duplex” connection achieved in between managers is done with SOAP

webservices, this connection can be considered the trunk connection between managers.

Last but not least the connections to the database (using MySql) are made through JDBC, this technology permits to have the object oriented functionality on a persistent context.

The use case diagram is represented in Figure 2. At the final stage “addMailbox” and “removeMailbox” in MailBoxManager are invoked by “addUser” and “delUser” in DirectoryManager respectively. And the “authenticate” use case in DirectoryManager is invoked by “logIn” in the MailBoxManager. So every time you add/delete a user a mailbox associated to that user is created/deleted and the login helps to maintain a session during the execution of several actions from a client so he/she will be able to manipulate only his/her mails.

Figure 3 and Figure 4 presents the Directory and Mailbox package classes respectively.

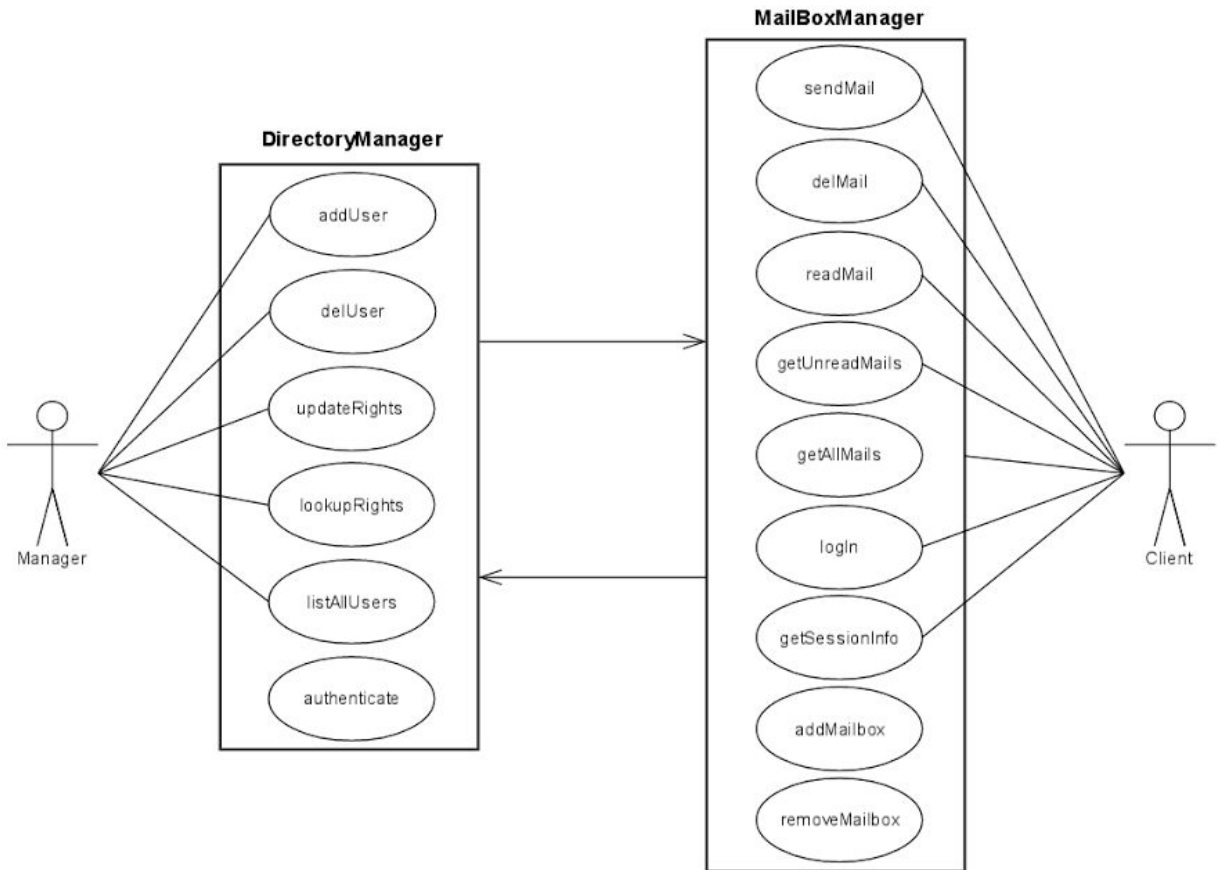

Figure 2: Use Case Diagram

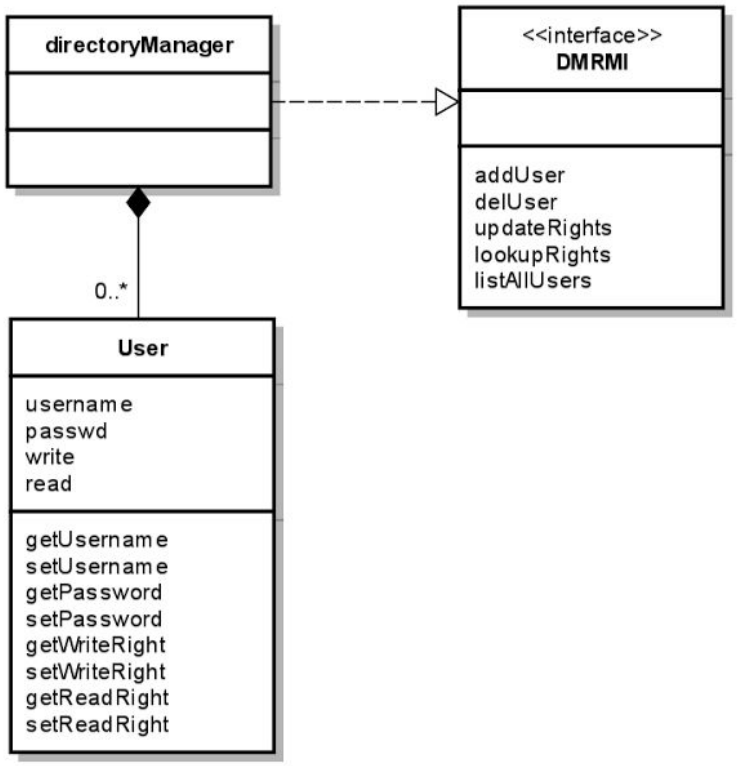

Figure 3: Directory Class Diagram

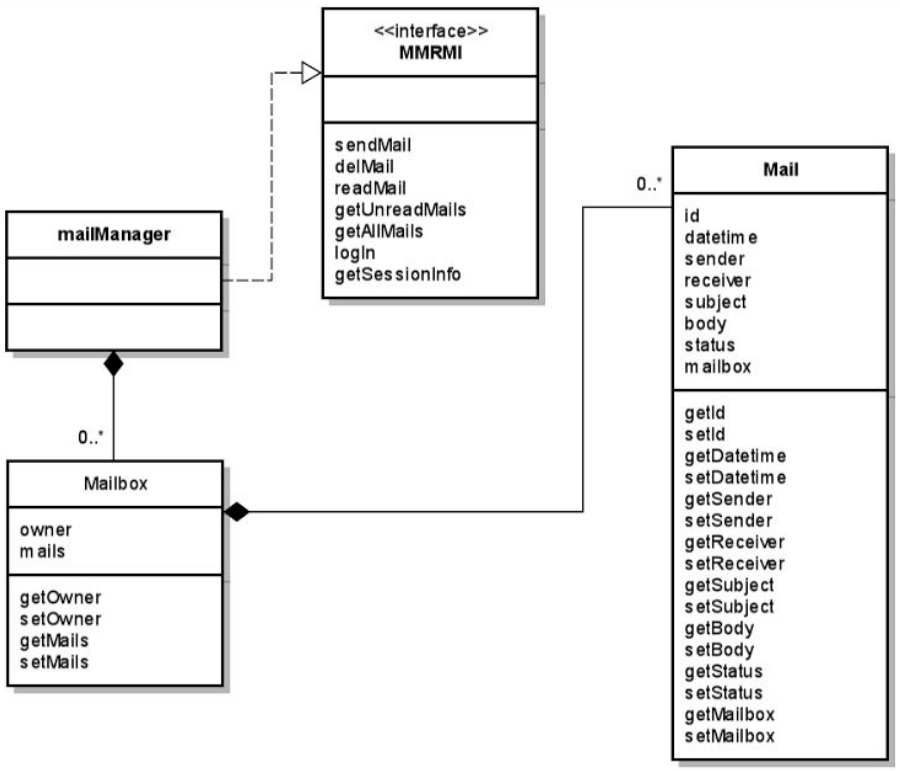

Figure 4: Mail Class Diagram

Some design patterns that we take in consideration while doing this project are shown in figures:

- Figure 5: Facade or also known as Wrapper or Adapter - Interface transformation, a design pattern for distributed interaction that is used in the context where there are clients requesting services, servers providing services and services defined by interfaces.
- Figure 6: Factory - Entity creation, also a design pattern for distributed interaction where applications are organised as a set of distributed entities.

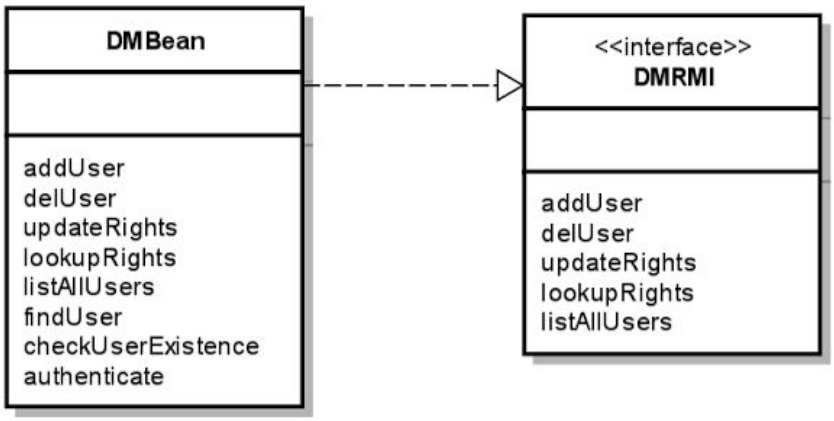

Figure 5: Facade Design Pattern in Directory Manager

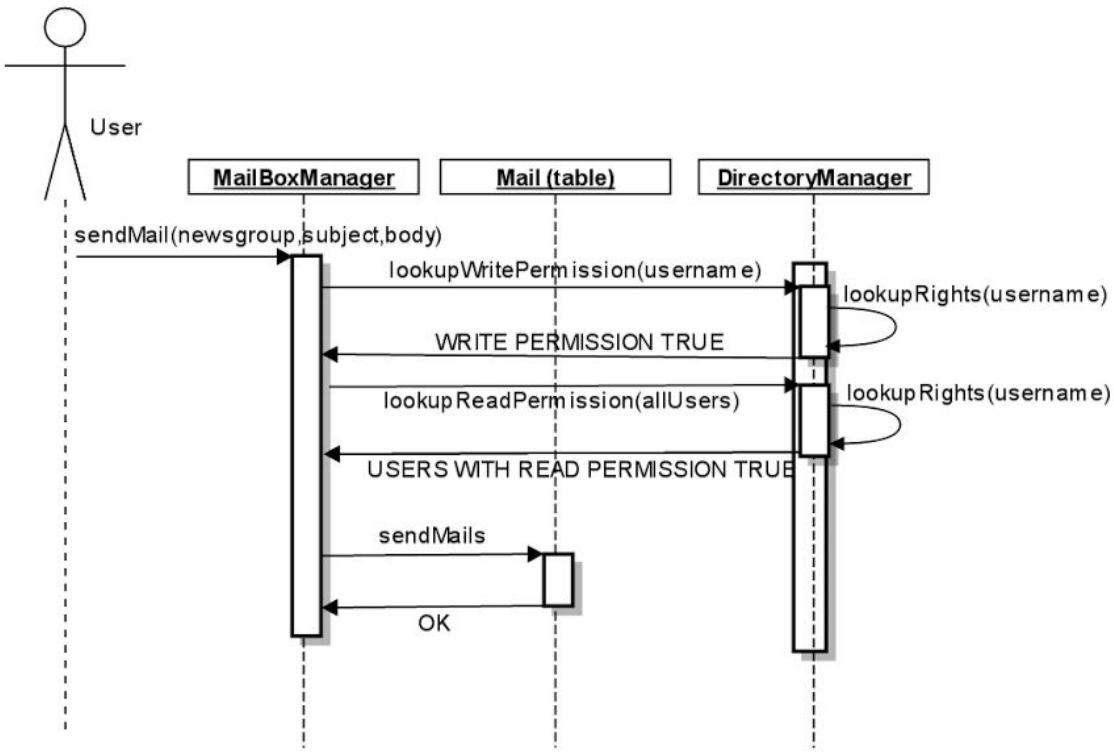

Figure 6: Send a message to a newsgroup Sequence Diagram

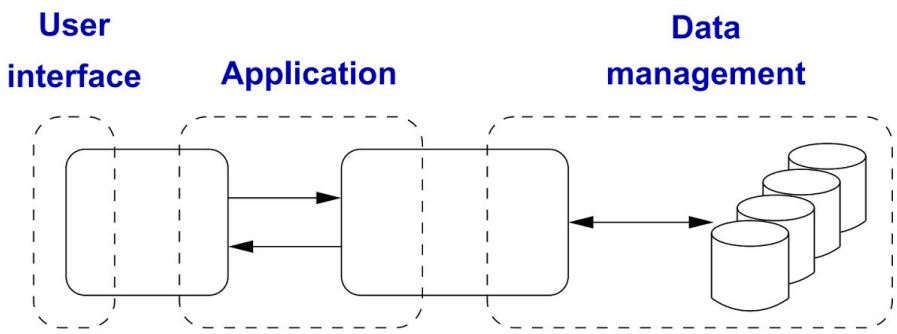

Figure 7: Multi-tier Architecture - Horizontal Decomposition

- Figure 7: Multi-tier Architecture a design pattern for composition, specifically horizontal decomposition separating system functionalities when a complex distributed system is presented.

How to Run

The main project folder will contain two subfolders, mailManager and directoryManager. Each folder contains a runme script, this script is able to detect the machine ip (fedora, using wireless interface) and also will prompt for the second server ip, in this way both machines will be able to perform a wsimport on each other. Since both servers depend on each other webservices to generate the classes we need to false start one server. This simply means to start one server wait until is up, start the other server wait until is up and finally restart the first server. If you want to run both servers in the same computer make sure to input the actual ip address not localhost or 127.0.0.1.

The runme script are prepared for this starting sequence so only make sure to follow the script instructions to let the servers connect. After this the script will automatically start the clients which will run the test cases. After the test cases are over the client will allow interaction through both clients mail clients and admin client. With the interaction menu you will be able to perform every kind of operation to test if the system responds correctly to the inputs.