Micro Project

Sophie Chabridon, Chantal Taconet

Revision : 545

CCN/CSC7321

Télécom SudParis—INF

octobre 2015

1 Introduction

The aim of the project is to design and realize a mailbox manager. The mailbox manager handles classical 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 different mailboxes. Only some users have the right to send messages to the common newsgroup.

Architecture The general Architecture of the system is presented in Figure 1.Clients and servers are distributed over the network.

- A *Mail Box Manager* manages the mail boxes. There is one mail box by user of the system and one news-box shared by all the system's users.
- A *Directory Manager* manages the users and their access to the system. In the final version of the system, the *Mail Box Manager* verifies with the *Directory Manager* the user rights concerning the access to the common news group.
- A *Mail Box Client* accesses the *Mail Box Manager* services.
- An *Administration Client* accesses the *Directory Manager* services to add, remove and modify rights of the users.

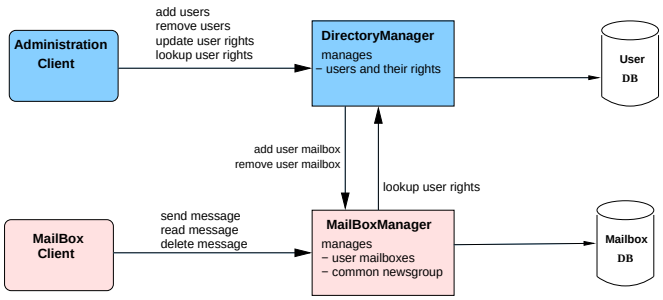

Figure 1: General architecture of the system

2 Project UML diagrams

An analysis of the problem (perhaps not your final analysis, you can make different modeling choices), has given the following diagrams.

Use Case Figure 2 presents the use case diagram. At the final stage, the "add mailbox" use case may be invoked by the "create user" use case of the "Directory manager" (the same for respectively "remove mailbox" and "delete user").

Directory class diagram Figure 3 presents the Directory package class diagram.

Mailbox class diagram Figure 4 presents the *Mailbox* package class diagram.

Send a message to a newsgroup sequence diagram Figure 5 presents the sequence diagram of the use case send a message to a newsgroup in its final stage (*i.e.* with access rights verification).

3 Middleware technologies involved in the project

- The *Directory Manager* and the *Mail Box Manager* are developed using *JavaEE* technology. All the mailboxes, the newsgroup and the user rights are persistent.

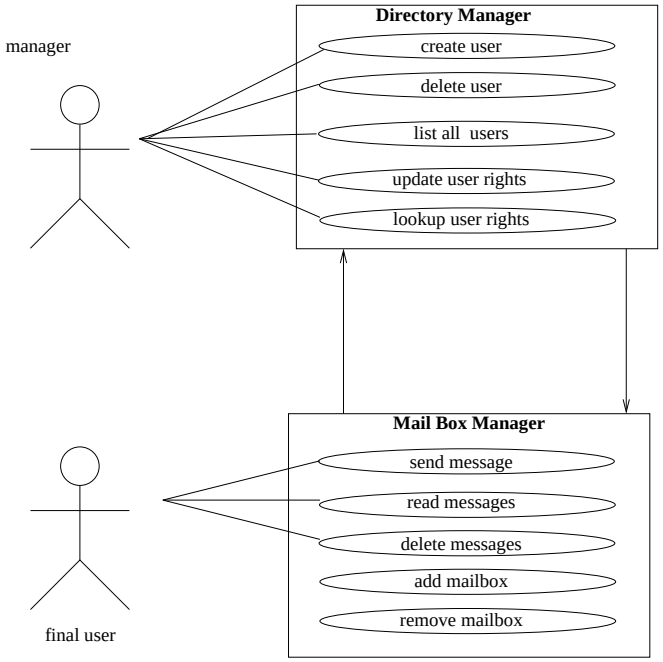

Figure 2: Use Case Diagram

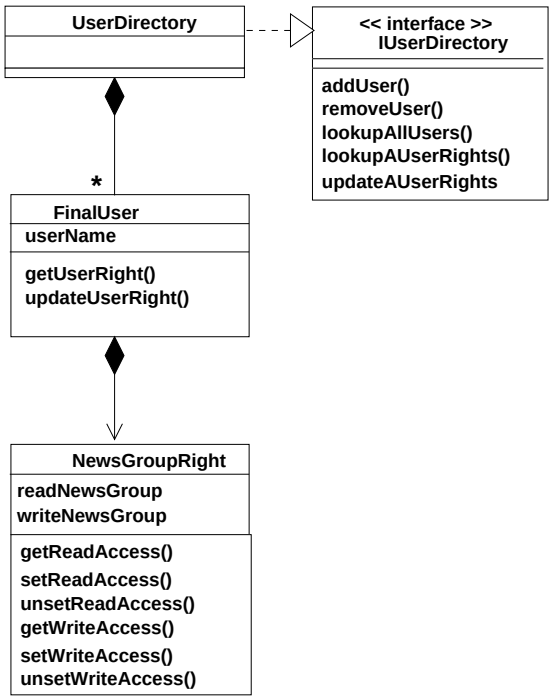

Figure 3: Directory Class Diagram

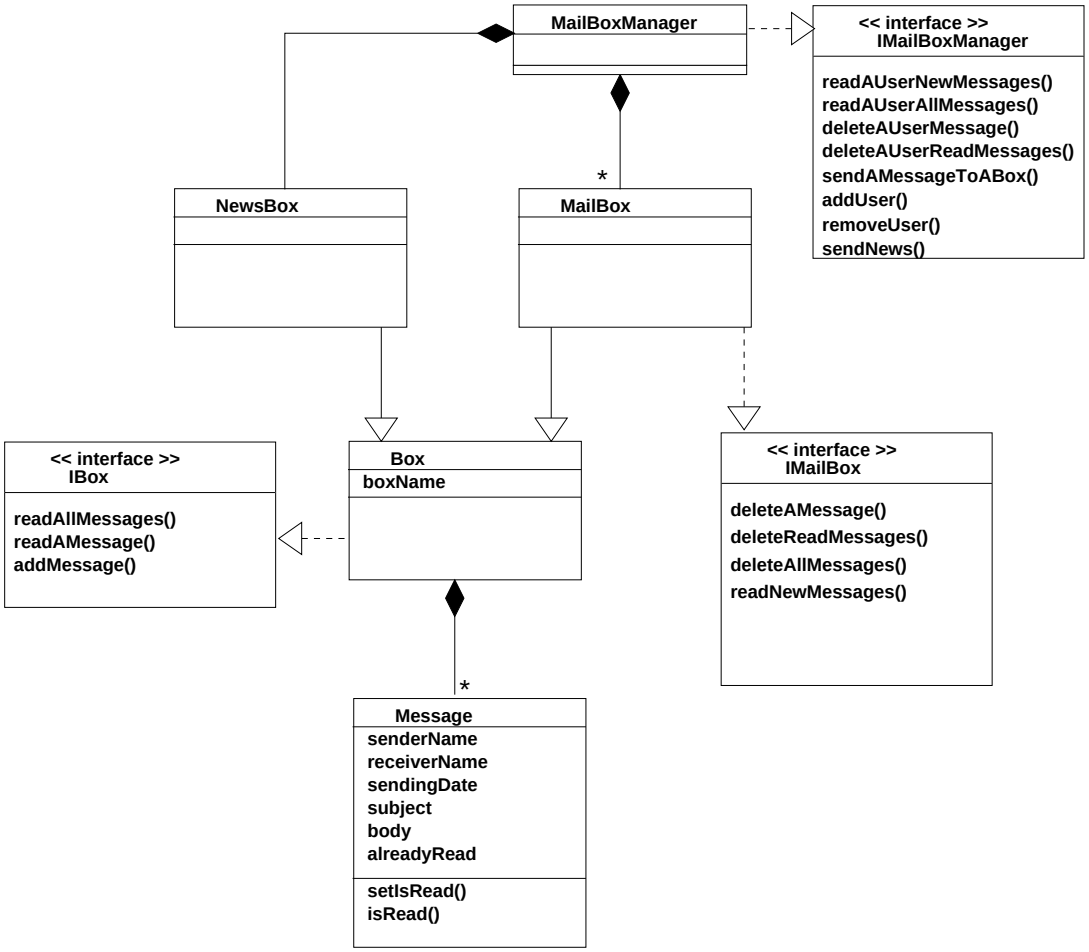

Figure 4: Mailbox Class Diagram

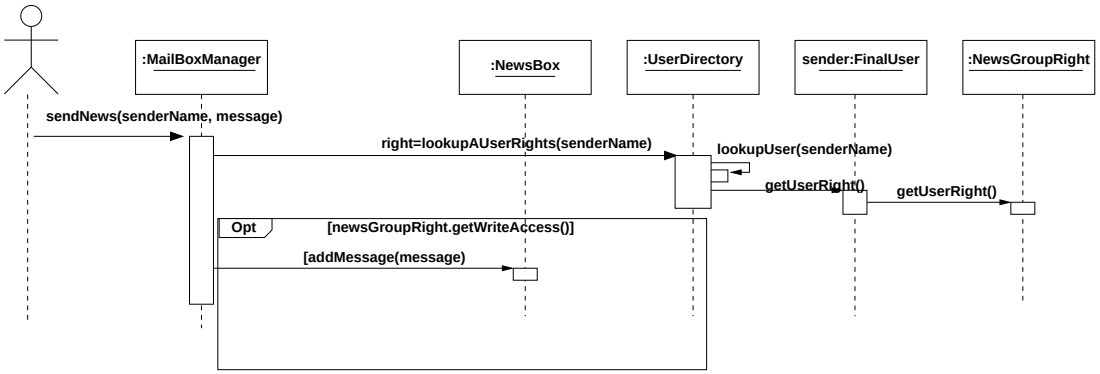

Figure 5: Send a message to a newsgroup Sequence Diagram

- The communication between the clients and the Managers will be realized through Java RMI for the administration server and Web Services for the mailbox manager.

4 Micro Project steps

The project will be realized in several steps.

1. Architectural choices :
   - With the material in the course "Introduction to middleware through design patterns", identify the design patterns involved in the application. For each design pattern, draw a picture presenting it and give adequate explanations. These pictures have to be included in the final report.
   - Design the persistent data structures: Do not use SQL keywords in the description of these data structures. For instance, *user* and *right* are reserved SQL keywords and should not be used in your own data structures.
   - Describe the facade of the services: define its interfaces.
2. Design and implement the *Directory Manager* and the *Administration Client*. The *Administration Client* is a test client with a predefined scenario.
3. Design and implement the *Mailbox Manager* and the *Mailbox Client* (without access rights verification). The *Mailbox Client* is a test client with a predefined scenario.
4. Add access rights verification to the *Mailbox Manager*.
5. Distribute server and clients on different computers.
6. Add a RESTful interface to at least one of your application components and develop a client to test this interface.
7. Write a script to automate the distributed demonstration (NB: you can use ssh to start a command on a remote computer: *ssh hostname "cmd"*).

Persistency of mailboxes The application may be started and stopped several times. It is therefore required to have persistent mailboxes and user directory whose content is saved in a database. You must use the same *MySQL* database as during the labs and administrate it using the *phpMyAdmin* service.

Clients and user interfaces This micro-project is about middleware for distribution not about user interfaces! We ask for simple clients, *i.e.,* test clients with no interactions and with predefined users and messages are absolutely sufficient. You are asked to write either test clients with predefined messages or command line clients (to specify all the parameters on the command line).

5 What is expected at the end of the project

The results of the micro project will be:

Report A small report (max. 10 pages) describing your solution and the associated design patterns, the architecture and technology choices, the persistence solution, a short user manual so that we can test your project, the encountered difficulties and all information you will judge necessary.

Defense An oral presentation of 15 minutes including a demonstration will allow you to describe the main design choices you made and to show your work in action. Both partners of the team are expected to present some part of the work.

Demonstration A demonstration showing your solution in action. Your implementation has to be original. Some variants on the subject are encouraged, UML diagrams are provided on the subject only as examples of a design solution. **Develop step by step in order to have an operational demonstration**.

Sources All the material (report, slides of the oral presentation, program SOURCE files) has to be uploaded in moodle before the deadline and in one archive. The archive will contain a root directory with the name of the students who worked on the project (*e.g., nameA-nameB.tgz* archive contains *nameA-nameB/* directory). Be careful :

- Do not include files that can be generated by ant (e.g. .class);
- Use ascii7 characters only in the name of files (no accent, no white etc.);
- Include the *build.xml* (or the *makefile* or the *pom.xml*) used to automate the generation of classes and also to launch the demonstration;

- If you develop your application with the Eclipse Integrated Development Environment (IDE), make sure that it can be run as a standalone application;
- Provide a good quality code verified with *checkstyle* and *findbugs*.

Good work!