UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEP. ENGENHARIA INFORMÁTICA	**Distributed Systems** **1st Semester 2013-2014** **PROJECT #1** **Deadline: 25-Oct-2013**

IdeaBroker – Idea Management and Trading

Objectives of the assignment:

At the end of this practical project, you should have:

* Implemented an Idea Management System, following a client-server architecture. It should use TCP/IP sockets for communication, and the server should follow a multi-threaded model;
* Created a second communication layer taking advantage of the JavaRMI higher level API.
* Ensure the availability of the service, by implementing a fail-over solution.

Description of the Work

Ideas are a powerful concept. The way in which they are organized in the Web is not the best, as it is seldom structured. This project aims to organize organize ideas against each other, and to provide users with an overview on a specific topic as well as an indication of how much each idea is worth.

In order to promote diversity in the projects, each group should pick a different theme for their idea manager. Some ideas include politics, sports, hobbies, technology, computer science, problem solving, mathematics, etc. Users should be able to exchange and discuss ideas, as well as to trade ideas much like in a stock market.

Functional Requirements

Your application should give two options to every user that connects to it: register an account, and login. After logging into the system, there are several options you should provide the user with:

(1) **Create a Topic:** Topics work as themes for ideas to be connected to.

(2) **Submit an Idea:** An idea should be connected to (at least) one topic. An idea can also be connected to other ideas, and that relationship can have a neutral, agreeable or disagreeable nature. Ideas can also have attached files that should be uploaded to the server.

(3) **Delete an Idea:** If an idea is added by mistake, the author should be able to remove it. However, that is only possible if the user owns all the shares of the idea.

(4) **Buy shares of an idea:** When an idea is submitted, 100% of its shares belong to the author. Other users can buy X% of that idea instantaneously, without any need for approval from the seller.

(5) **Set shares selling price:** Users may increase or decrease the price at which their shares can be automatically sold. This price is mandatory upon buying shares, or creating an idea.

(6) **Show transaction history:** Each user should be able to see their last X transactions.

(7) [Groups of 3 only] **Create and manage private group:** Groups are a feature mandatory for teams with 3 members. Both Topics and Ideas can be connected to a group, making them restrict to group members. Group managers should be able to add and remove users from groups.

Non-Functional Requirements

(1) **Clients should not be aware of network problems, except for long network failures.**

(2) **Data should be persistent**. This means that crashes and network problems should not compromise the existing data, either on the client or the server.

(3) **Ideas in progress should not be lost**. Given the importance of ideas, you are not allowed to loose any of them, even if they are not committed to the server.

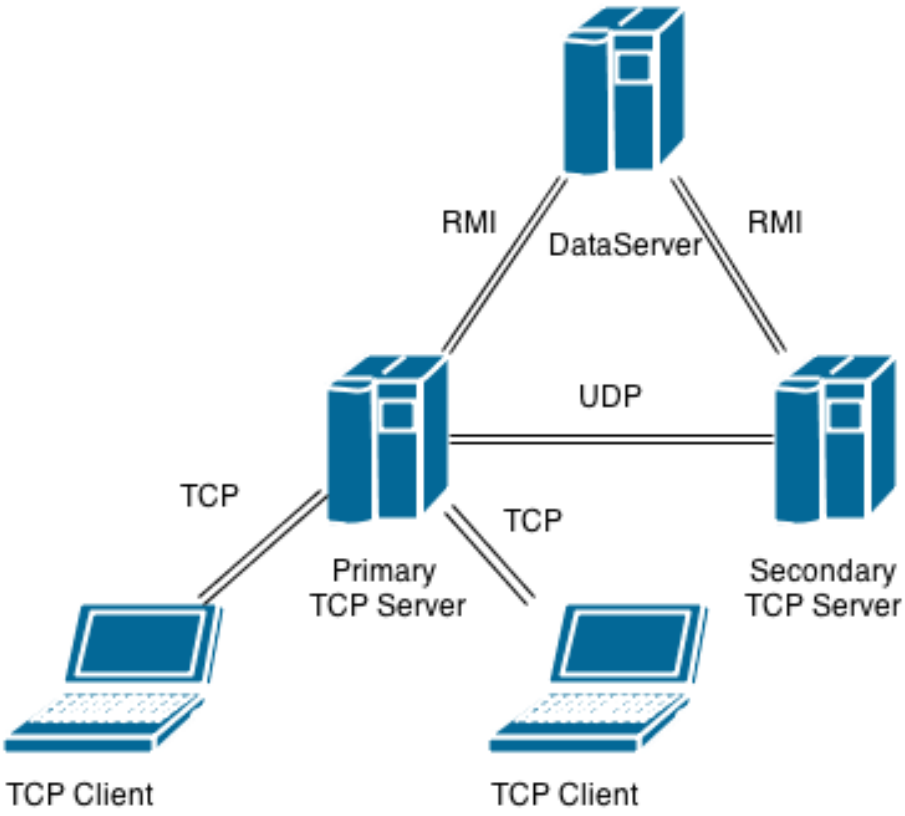
(4) **Strict order of buying/selling idea shares, even in hand-over scenarios.** Even when the primary server fails, clients that submitted an order before clients that submitted on the secondary server should still have priority when buying shares. Selling is first-come-first-serve.

(5) **Share trading should be transactional**. If there is some network problem, a trade should be either fully completed or fully reverted. You are not allowed to remove the shares of someone without depositing in someone else's account. Each new user starts with 10000 DEIcoins, so the total number of DEIcoins in the system should always be N * 10000, being N the number of registered clients.

(6) **Separation of concerns:** Your communication, business logic and data should be separated and should be able to be deployed to different machines.

(7) **Any User Interface is acceptable.** Graphical User Interfaces will not be rewarded on your grade, so you should focus on the contents of the project, robustness, availability, fail-over, communication protocols and not on prettifying your UI.

(8) **Demo has to be performed on 2 or 3 computers (one per team member).**



Recommended Steps

1 - Architecture

First of all, you should design the architecture of your solution. You should have an overview of the components (server, client, database) and how they connect to each-other (TCP, UDP, RMI). Note that although you might run several components on the same machine during development, you should consider all components running in different machines in deployment. You should validate your architecture with the teaching staff before advancing to the next steps.

2 – Data Structures

You should now define how to represent your data, both in-memory and on disk. You should define the interfaces and classes that will hold **Users, Topics, Ideas, Relationship between ideas,** and **Shares.** You should also decide which persistence solution to use (binary file, text file, sqlite, mysql, postgres, oracle, mongodb, etc…)

3 – TCP Client-Server

You should now create two applications. The first is a client application for users to execute. That application should receive commands from the user, send them to the server and print the response. The other application should be a server that should be able to receive commands, modify the data-structures and reply to the client. You should focus on the availability and fail-over aspects of TCP communication, handling all possible problems in the networking stack.

4 – RMI Client-Server

You will build a RMI-Server that will wrap database operations to provide that operations over the network. You will then implement a RMI Client on your TCP Server to access those operations. The goal is to separate business logic and models from the communication logic. You should also focus on handling all possible errors in the best possible way. Note that the RMI Server may store data into files or an actual database to guarantee persistence.

5 – UDP-based Fail Over

Machines also fail. Corrupted RAM, failing hard-drives, burned BIOS, etc…. In your deployment, your TCP-Server application should be executing in two different machines, being one of them the primary. If that machine stops to provide the service, the secondary machine should take over and they switch roles. If the failing machine recovers, it should now act as secondary.

6 – Report

You should take time to write a report at the end of the project, considering the previous steps. You should write the report as documentation so a new developer may come in and understand how your solution was built, your technical decisions, and may introduce new components and modify or replace existing ones.

Report:

You should include the following sections in your report.

(A) Introduction
(B) Internal architecture of the service (focus on interoperability)
(C) Data model of the application
(D) Protocol-specific communication (TCP, UDP, RMI)
(E) Architecture of File Transmission
(F) Exception handling in sockets and RMI
(G) Fail-over solution
(H) Total order implementation
(I) Installation and configuration manual
(J) Description of the tests made to the application

What Will You Learn

With this project you will acquire practical competences in the following topics:

- Sockets programming in Java
- Exception handling in Java sockets
- Development of multi-threaded servers
- Java RMI
- Implementing a Fail-Over solution
- Implementing Server-code with support for multi-protocols

Future Phases of your Work

In the next assignment, you will have to enhance your service by:

* Creating a web-based version (HTML/JSP/JavaBeans)
* Integrating the application with a third party web API.

Assignment Upload:

Deliver everything in a `ZIP` file. This file should include a `README` file with all the INFORMATION NECESSARY to execute and test the assignment without the presence of the students. Assignments that do not contain this `README` with all the necessary instructions will not be evaluated. Assignments that do not execute correctly will also not be evaluated.

Also inside the ZIP file, there should be a PDF containing your report. Do follow the suggested structure, as all sections will be evaluated.

You should submit the ZIP on the inforestudante platform: http://inforestudante.uc.pt