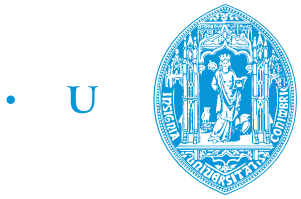
Project 3

Enterprise Application Integration
Department of Informatics Engineering
Delivery date: ver em submissão de trabalhos



Objectives

- Gain familiarity with message-based communication.
- Learn how to use the Java Message Service (JMS) 2.0 API.
- Create asynchronous loosely-coupled applications.

Final Delivery

- You must submit your project in a zip file using Inforestudante. Do not forget to associate your work colleague during the submission process.
- The submission contents are:
    - Source code of the requested applications ready to compile and execute.

The REPORT is expected to be complete in the sense that it needs to contain all necessary and sufficient information for the teacher to give the score to the evaluation item, even without having to run the code itself and without having to meet with the group in person for evaluation. For that you can include descriptions, screenshots, code extracts, whatever is needed for a complete and thorough evaluation. If the report is absent the score is 0, and if it is incomplete the score is significantly affected. This is to make sure you do have a complete report.

Before the main body, the REPORT starts with the complete identification of the students and group, then a table of contents, then the following:

a. Lists of what the group succeeded to do and what is missing
b. self-evaluation of the group (0-100%)
c. List of what each student contributed (no repetitions)
d. self-evaluation of each student in the group (0-100%)
e. hours of effort by each student separately

These items a,b,c are important for the teacher to check whether his evaluation coincides more or less with what the group and student thinks.

Project

Description

In this project, we will create a scenario that makes use of messaging to support interaction with your publication data from the previous project. In this new scenario, there are two nodes (user and system), which correspond to two types of applications that will be used by two different types of human users: **researchers** and an **administrator**. Notice that there is only one administrator, while we can have multiple researchers using the system. Since one of the key goals is to learn the JMS API, all communication between the different nodes will be supported by JMS. The functionality required is described in the next paragraphs.

User node (researcher)

Each researcher can perform the following operations (**note that this node does not have direct access to the database**):

- Register using a username and password. Registration must be approved by the administrator.
- Login in the researcher application, using a username and password. In case of success, the user will be allowed to use the remaining functionality of the application.
- List all publication titles in the system.
- Given a publication title, obtain all the associated data.
- Add a new publication (must be approved by the administrator).
- Update a publication (must be approved by the administrator).
- Remove a publication (must be approved by the administrator).

System node (administrator)

The system node is the only node that has direct access to the database. The system node allows the administrator to carry out the following functions:

- List all users.
- List all pending tasks (registrations; adding, updating, or removing publications).
- Approve or reject a pending task:
    - Registration: the user is notified of the outcome.
    - Adding publication: this will result in adding the information to the database and notifying all users.
    - Updating the information about a given publication. This will result in notifying all users.
    - Remove a given publication. This results in notifying all users.
- Deactivate a user. This disallows the user to continue using the system (i.e., any further requests by that user will be rejected. It must also be possible to re-activate a user.
- List all publication titles in the system.
- Show detailed information regarding a publication.

Final remarks

- All incoming messages must have an immediate on-screen print.
- You must use JPA if database access is required.
- Depending on your architectural choices, you might want to run code in an additional Thread. You can see how to use one at:
  http://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html
- Some of the functionality has been simplified to shorten the assignment. Make sure you identify which points should be built differently in a real system.
- As you may notice, the description of the assignment is intentionally left incomplete. You are expected to be able to make sensible implementation choices at specific points. Refer to your Professor for any architecture/functionality doubts.

Good Work!