Assignment 3
Analysis and Design Document

Dan Fulga
Department of Computer Science
Technical University of Cluj-Napoca
Group 30432
danfulga96@yahoo.com

May 10, 2018

Contents

1 Requirements Analysis

1.1 Assignment Specification

The task of this assignment is to design and implement a client-server Java application for a news agency.

1.2 Functional Requirements

The application should have two types of users: readers and writers. Only the writers have to provide an username and a password in order to use the application, since their accounts are presented by the application developer and cannot be altered.

The readers can perform the following operations:
-when they enter the application, they will check as readers and proceed to see a list of articles
-click on the article they desire to read

The writers can perform the following operations:
-login in the system
-edit their own profile
-create, modify or delete articles

1.3 Non-functional Requirements

1.Data integrity

Data integrity is a very important requirement whenever dealing with an application that stores personal information. In this application, every data to be inserted is validated, and the system responds accordingly to invalid inputed data.

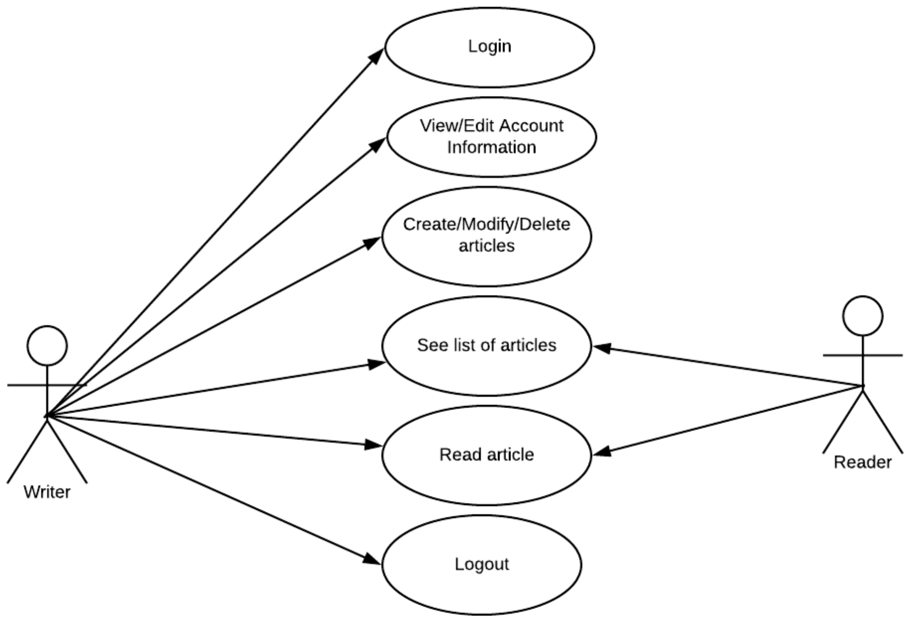
2.Extensibility/Manageability

This application can be easily further extended and due to its layered-architecture components, manageability is also increased.

3.Security

Considering that in order to view or manage certain data it is required to login using an existing account, the application is secured. Also, every role in the application(writer and reader) only sees what he is supposed to see.

2 Use-Case Model



Use Case: Update Writer Profile
Level : User-goal level
Primary actor: Writer
Main success scenario:
1.The writer introduces an username and a password and he is successfully logged in.
2.The writer will click on the appropriate button to update his profile.
The writer will insert data in all the required fields and click on the save button.

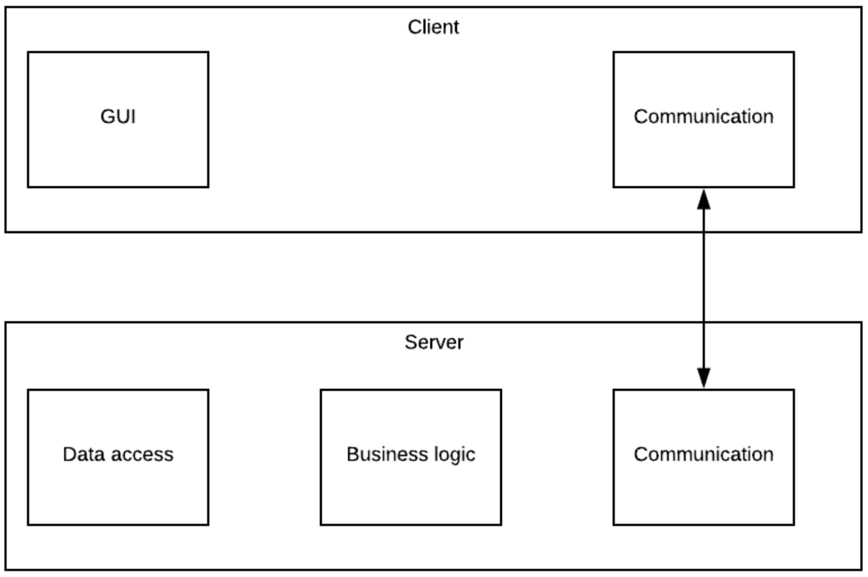
Extensions:

1. In case of login failure, the writer will see errors such as "user not found" or "invalid password" and his access to the application will be denied.
2. Data-related errors, for example invalid password such as: "Password format is not valid. Please satisfy the following conditions: a digit must occur at least once, a lower case letter must occur at least once, an upper case letter must occur at least once, a special character must occur at least once, no whitespace allowed in the entire string, anything, at least eight characters though".

3 System Architectural Design

The main architectural approach used for implementing this application is the Client-Server architecture. A typical client-server architecture consists of a server application running on one machine, and one or more clients – often, but not always, operating on different machines across a network. The server is modeled as a perpetual process that waits for clients to connect (request a service), and then processes the client requests. Java threads are ideal to the implementation of a server application, because a new thread can be started for each connecting client. This prevents one client request from holding up others if and when it enters a wait state, or some time-consuming task.

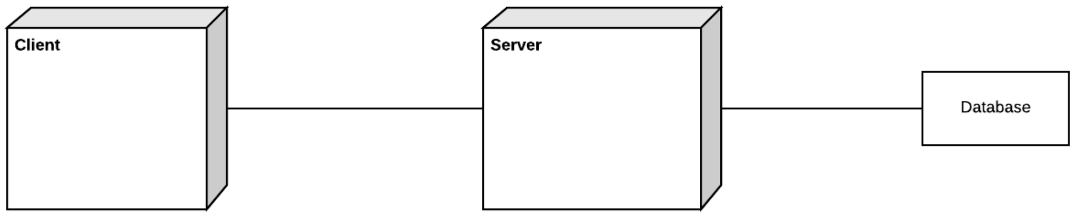
Sending the data between the client and the server is done using Json Serialization.



4 Diagrams

4.1 Package Diagram

4.2 Deployment Diagram



5 UML Sequence Diagram

6 Class Design

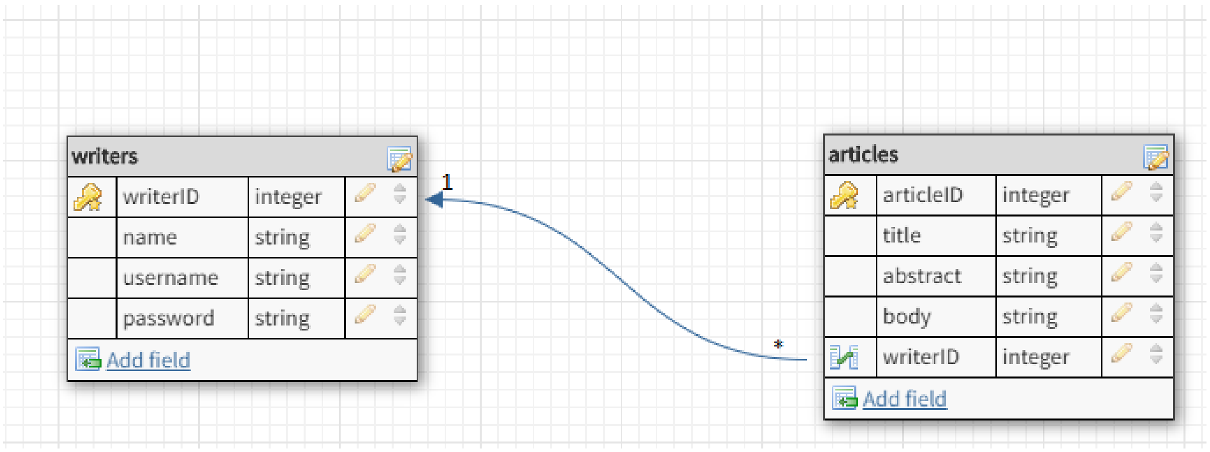
6.1 Design Pattern Description

The Observer Design Pattern

The observer pattern defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically. The object which is being watched is called the subject. The objects which are watching the state changes are called observers or listeners.
In this project, the subject being watched is a list of articles, and the dependents are the readers. In other words, every time a writer adds a new article, the readers will be notified.

6.2 UML Class Diagram

7 Data Model



8 Bibliography