Assignment2

Analysis and Design Document

Student: Álvaro Suárez del Cueto

**Group: Erasmus**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

# Application for the management of Reader/Writers.

# Functional Requirements

The application has 2 types of users: the readers and the writers. The **readers** can view a list of articles, read an article and do not need to login in order the use the application. The **writers** need to authenticate in order tocreate, update or delete articles. So the writer accounts are preset by the application developer and cannot be altered.

An article has the following components:

* Title
* Abstract
* Author
* Body

The application must support multiple concurrent users. If a writer posts a new article, the readers must see it in the list of articles in real time, without performing any refresh operation.

# Non-functional Requirements

# 1.3.1. Availability.

* Source of stimulus: Writer.
* Stimulus: Access the login webpage.
* Environment: runtime
* Artifact: Client System.
* Response: the system ask for a pair username/password which will uniquely identify the writer.
* Response measure: 200 OK is the response if user exist, 404 not found if not

## 1.3.2. Performance.

* Source of stimulus: Writer.
* Stimulus: write article.
* Environment: runtime
* Artifact: Client System.
* Response: the system receives the fields needed to write the article.
* Response measure: The update of the information should be less than 2 seconds.

## 1.3.3. Security.

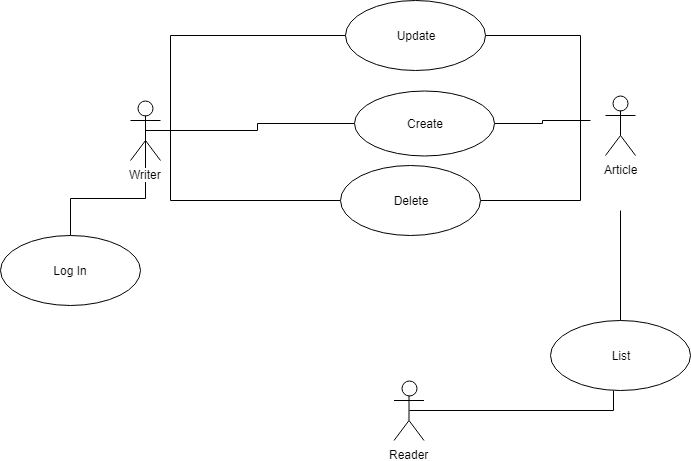
* Source of stimulus: Writer.
* Stimulus: Access the login webpage using an account.
* Environment: runtime
* Artifact: Client System.
* Response: the system ask for a pair username/password which will uniquely identify the writer.
* Response measure: if the writer does not exist system won’t access to writer main view and will show an error message.

## 1.3.4. Usability

* Source of stimulus: Writer.
* Stimulus: write article.
* Environment: runtime
* Artifact: Client System.
* Response: the system receives the fields needed to write the article.
* Response measure: webpage design and navigability should be easy for non-informatics familiar people.

2. Use-Case Model

* Use case diagram:



***Use case: Create.***

***Level: user-goal level.***

***Primary actor: Writer.***

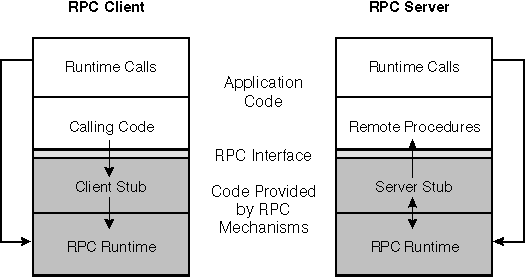
***Main success scenario: Writer creates an article.***

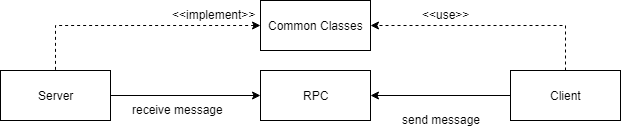
***Extensions: in case of success he will be notify with a success message, otherwise an error one.***

3. System Architectural Design

**3.1 Architectural Pattern Description**

The system will follow the client server architecture + rpc (Remote procedure call).





The server will handle the database and will keep it connected, also it will provide the interfaces for the rest of the systems.

It will exist an intermediate package which will contain all the classes that represent the entities and all the interfaces (in our case the repositories) that the server will use (explained below), the client and the server will have access to what we define in this intermediate system.

The RPC system will provide the mechanism to allow the communication between the client and the server (such as the message class), the steps to allow the client to use an interface defined by the server are the following:

* Define the interface with the corresponding methods in the common accessible part.
* Define the classes implementing that interface in the server class.
* In the server class define which classes correspond with which interfaces.

Example:

*IWriterDao wD = new WriterDao();*

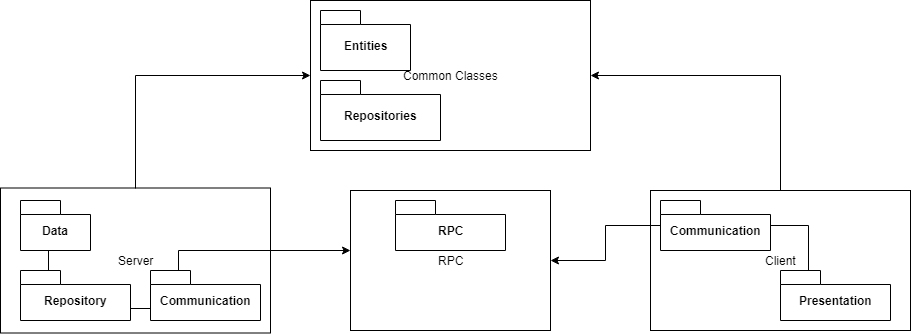
Registry.*getInstance*().registerEndpoint("IWriterDao", wD);

* The Registry class is used by the server to specify which object can be remotely used by a client.

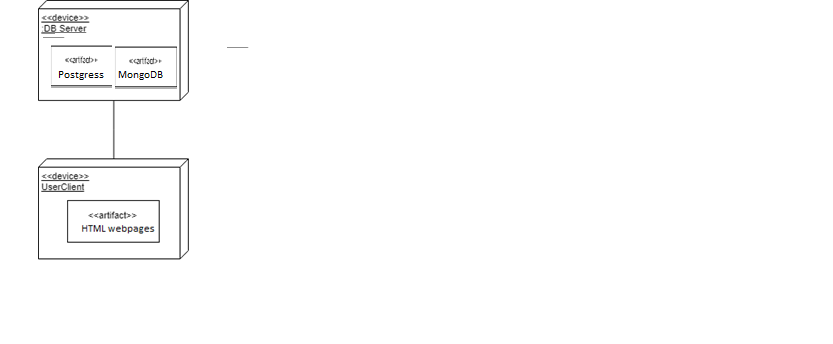
The client will contain the interfaces which will be shown to the final user.

**3.2 Diagrams**

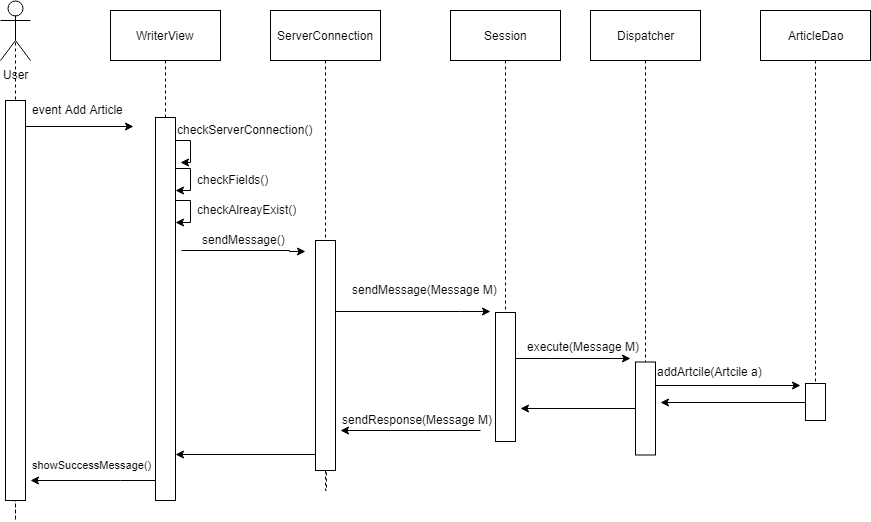
* Package:

**

* Deployment:



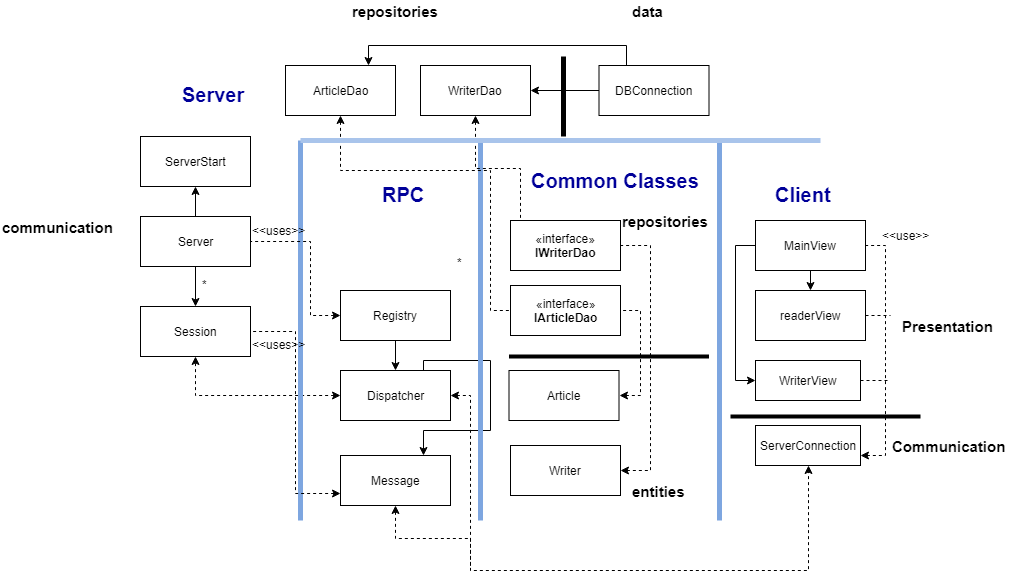
4. UML Sequence Diagrams



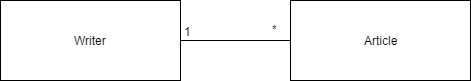
5. Class Design

**5.1 Design Patterns Description**

**5.2 UML Class Diagram**

**

6. Data Model

**

7. System Testing

The system will be tested using unit testing and using the dataflow method.