News Agency System for Readers and Writers Analysis and Design Document

Student:Ana-Maria Nanes
Group:30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implement a client-server application for a news agency. The application has 2 types of users: the readers and the writers. 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.

1.2 Functional Requirements

There are two types of users: readers and writers. The functional requirements of the system are defined for each of the user type, and also for the whole system.

The functional requirements are the following:

- The reader can perform the following actions:
  - ✓ view a list of articles
  - ✓ read an article
  - ✓ view all the writers

- The writer can perform the following actions:
  - ✓ register (create an account)
  - ✓ log in and log out
  - ✓ delete account
  - ✓ write an article
  - ✓ see similar (related) articles when writing an article
  - ✓ delete an article
  - ✓ modufy (update) and article of his own
  - ✓ view the list of all writers

1.3 Non-functional Requirements

The non-functional requirements :

- **Security** - making distinction between the two types of users – the readers do not have access to writers accounts and cannot modify the data kept about the articles in the database.

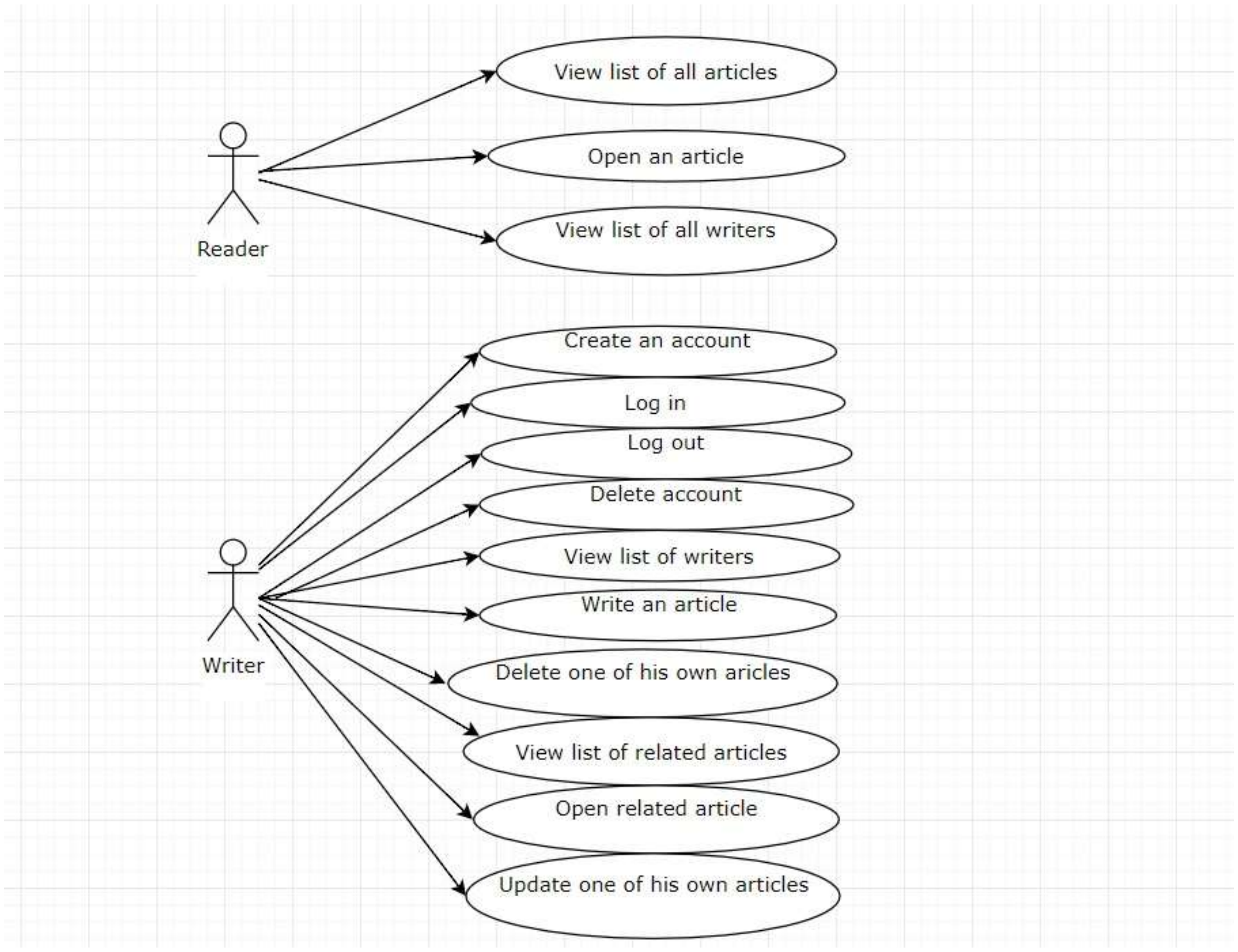
- **Data integrity** – is implemented using data validation each time new information is added to the system.

- **Extensibility** - due to the Layers Architectural Pattern it is easy to extend the system in more directions. And the concurrent acces of the clients has been implemented using threads and in the same way can be implemented the concurrent access of the writers.

- **BackUp** - the system`s information could be also stored in files – make a backup after a certain period of time

2. Use-Case Model

The following diagram is the use case for the system. There are two types of users: readers and writers.



Use case: Write an article
Level: Subfunction.
Primary actor: A writer
Main success scenario:

- Create a writer account by providing all the required information.
- Login by providing the correct password for the chosen username.
- Provide the information required for an article.
- Load the article in the system.

Extensions:

Scenarios of failure :

- the username is already taken.
- the password is not correct.
- the input data for the article is not valid – for example the abstract is not provided or the title is already used for another article.

3. System Architectural Design

3.1 Architectural Pattern Description

The architectural patterns used are the following:

1. Layers Architectural Pattern
2. Client-Server Architectural Pattern
3. The Model-View-Controller Architectural Pattern

1. Layers Architectural Pattern

This pattern can be used in order to structure the application such that it can be decomposed into groups of subtasks. Each layer implements subtasks at a particular level of abstraction and each layer provides services to the next higher level.

The following 3 layers will be used in order to offer the application the accurate structure:

- Presentation Layer
- Business Logic Layer
- Data Access Layer

These layers are modeled using different packages inside the application.

2. Client-Server Architectural Pattern

The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

In the case of the application for the News Agency the client-server architecture is implemented using sockets and the data that is transferred is in the json format.

Java Socket programming is used for communication between the applications running on different JRE, but in the case of this application they run on the same JRE. Also, in the case of this application Java Socket programming is connection-oriented. Socket and ServerSocket classes are used for connection-oriented socket programming .

The client in socket programming must know two information:

✓ IP Address of Server, and

✓ Port number.

3. The Model-View-Controller Architectural Pattern

Model–view–controller (MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

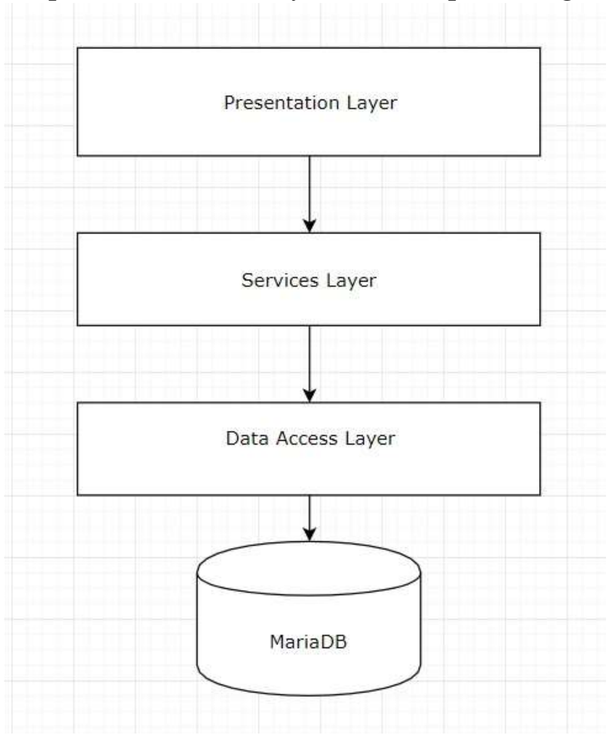
- Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- View - View represents the visualization of the data that model contains.
- Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

3.2 Diagrams

❖ **The system conceptual diagram:**

1. Layers Architectural Pattern

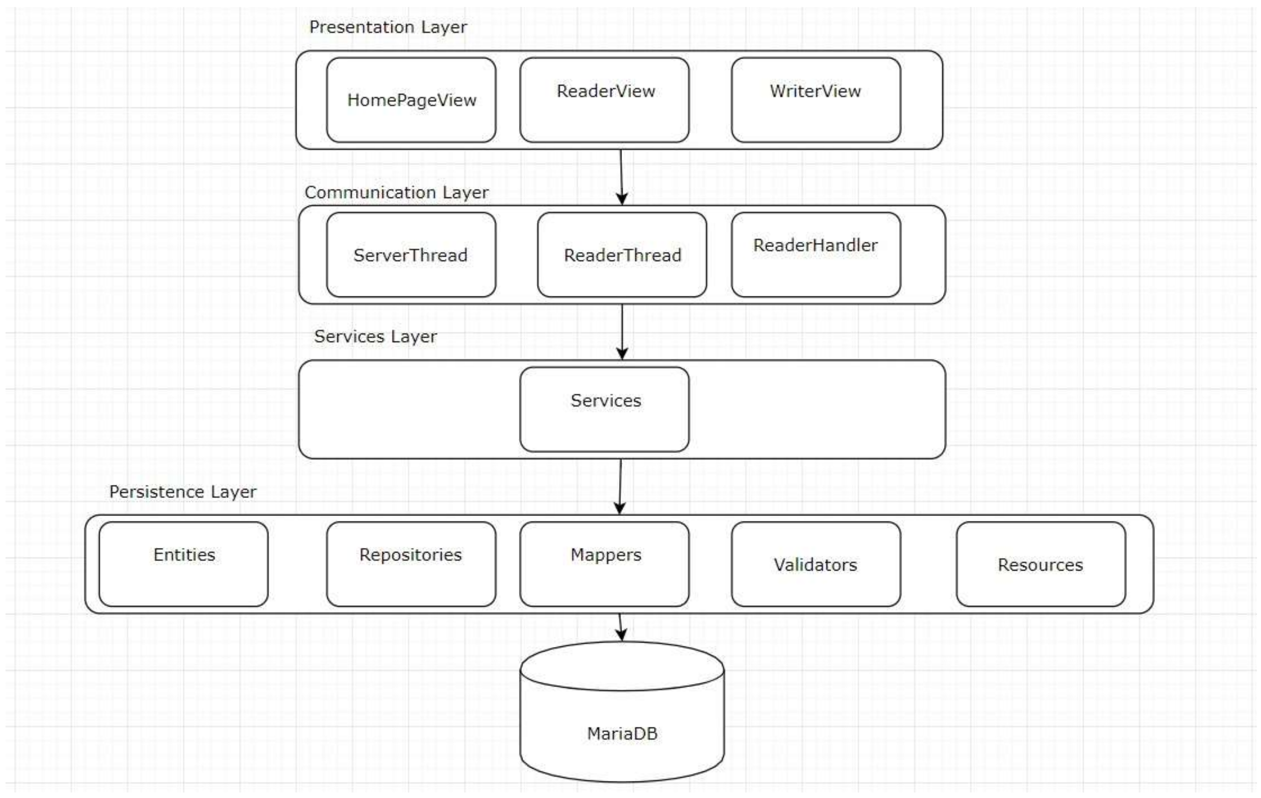
The layers structure is depicted in the next system conceptual diagram:



How the Layered Architectural Pattern is applied:

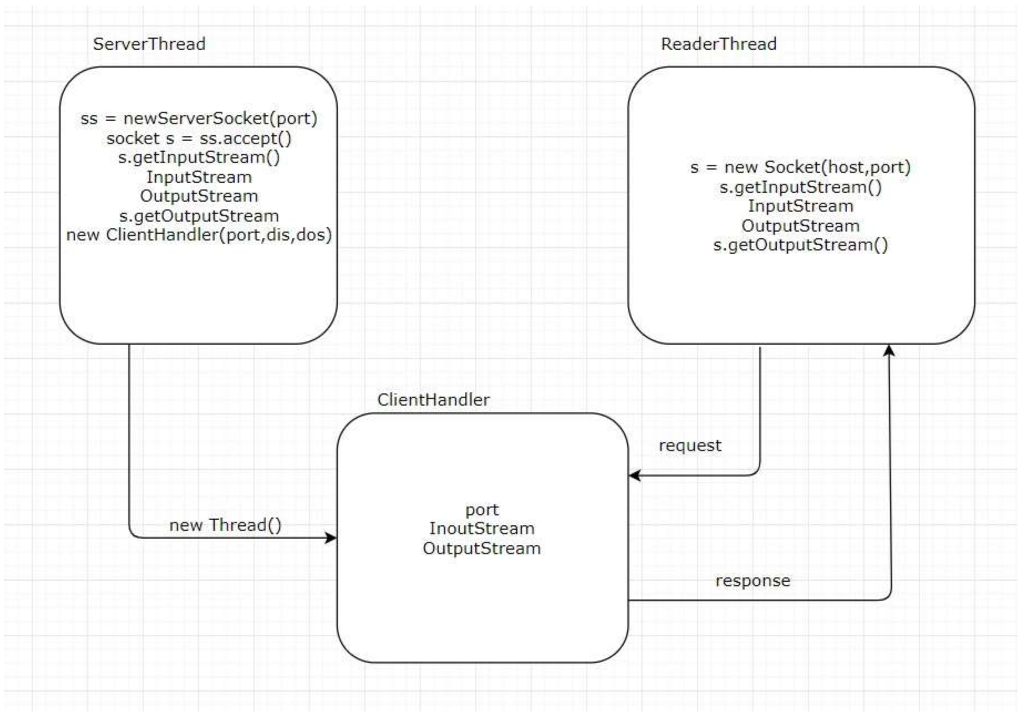
- ✓ The Data Access Layer provides access to the data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services.
- ✓ The Services Layer implements the core functionality of the system, and encapsulates the relevant business logic. It consists of components which expose service interfaces other callers can use.
- ✓ The Presentation Layer contains the user oriented functionality responsible for managing user interaction with the system,and generally consists of components that provide a bridge into the core business logic encapsulated in the business layer.

A more detailed overview is the following, in which we can see the classes that can be found in each layer:



2. Client-Server Architectural Pattern

The Client-Server Architecture is implemented in the Working communication Layer. In This layer modelled as a package we have the classes that are responsible for the client-server communication. In the specific case of this application we also create a ClientHandler mechanism – modelled as a class that extends the Thread superclass in order to allow the server to receive requests from all the clients and to send back the appropriate response to all of them.



How the Client-Server Architecture is applied using Sockets:

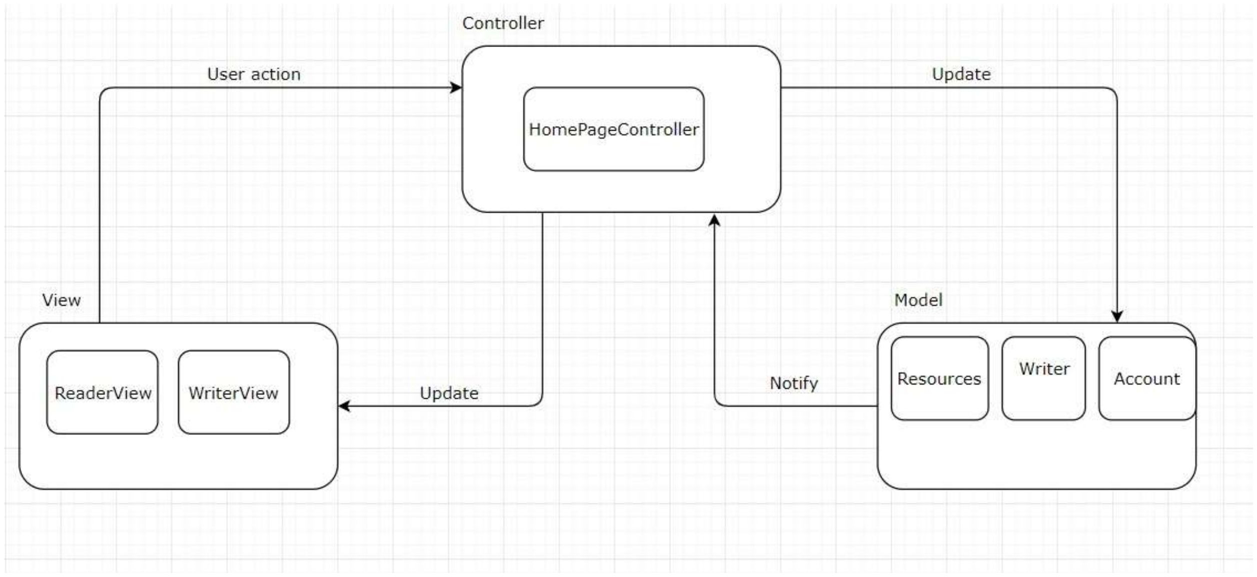
ServerThread class is responsible for:

- ✓ Establishing the Connection
- ✓ Obtaining the Streams
- ✓ Creating a handler object
- ✓ Invoking the start() method

ReaderThread class is responsible for:

- ✓ Establish a Socket Connection
- ✓ Communication

3. The Model-View-Controller Architectural Pattern



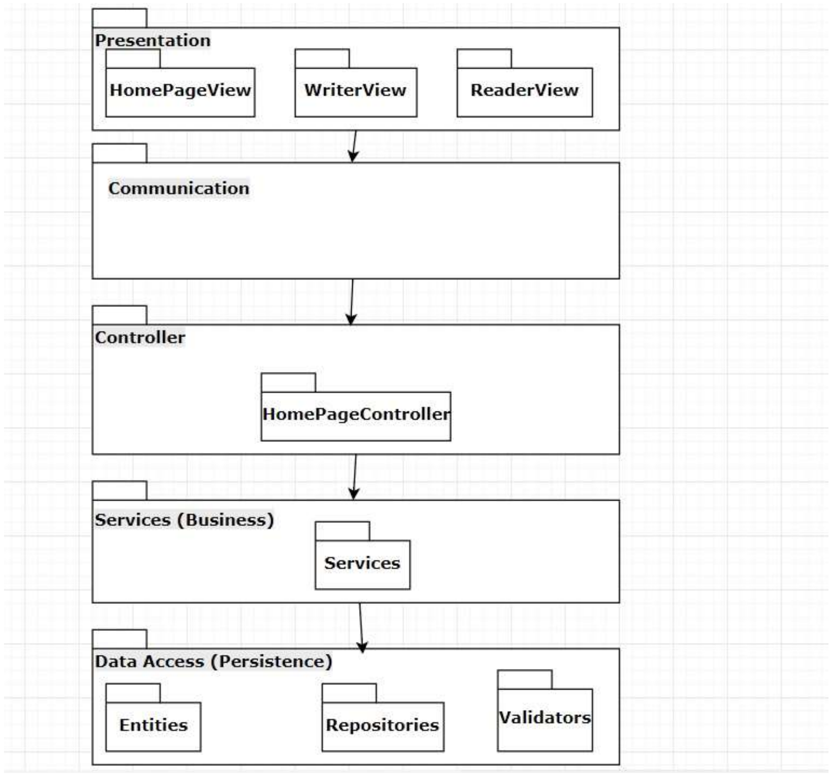
How the MVC Architectural Pattern is applied:

- ✓ The MODEL: consists of the classes that represent the entities from the databasses and that hold the required data to be processed. These classes can be found in the persistence Layer.
- ✓ The VIEW: is modeled in the presentation layer and consists of the classes that are used to create the GUI.
- ➢ The CONTROLLER: consists of the classes that are used to separate the view from the model and that control the flow of data.

❖ **The package diagram**:

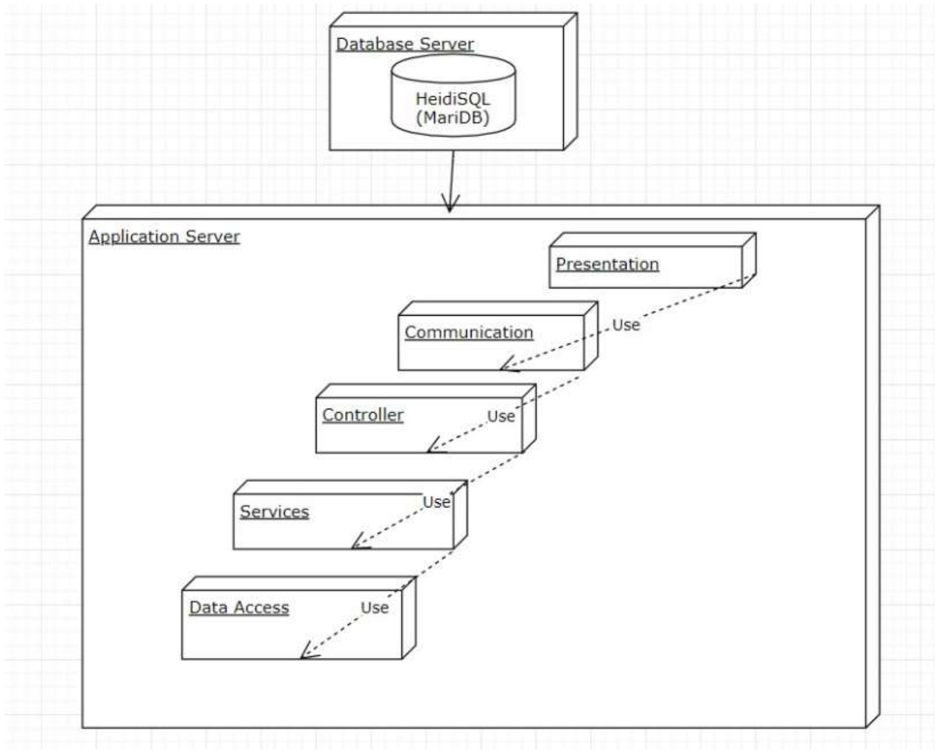
The Layered achitectural pattern is implemented using a separation of the application classes into packages according to their functionalities.
In the application we have the following packages:

- Presentation
- communication
- controller
- services(business)
- data access (persistance)



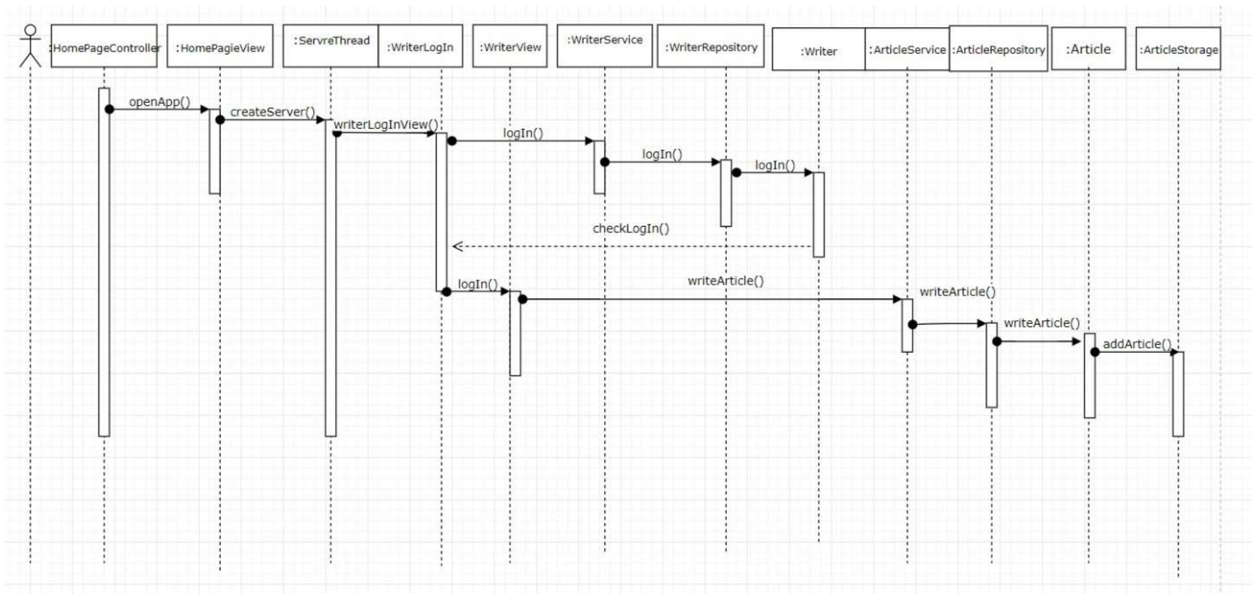
❖ **<u>The deployment diagram</u>**:

The deployment diagram is a structure diagram through which we see the architecture of our application distributed through artifacts. The artifacts represent physical elements from the real world that are a result of a development process. In our application, we only have 2 components, i.e the Database Server and the app itself, which is a desktop application. The principal elements of the deployment diagram are the nodes.



4. UML Sequence Diagrams

The following sequence diagram depicts the use case of writing a new article, action that can only be performed by a writer.



5. Class Design

5.1 Design Patterns Description

For implementing the functional requirements of the News Agency Management System the OBSERVER design pattern has been used.

<u>Observer Design Pattern</u>

The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.

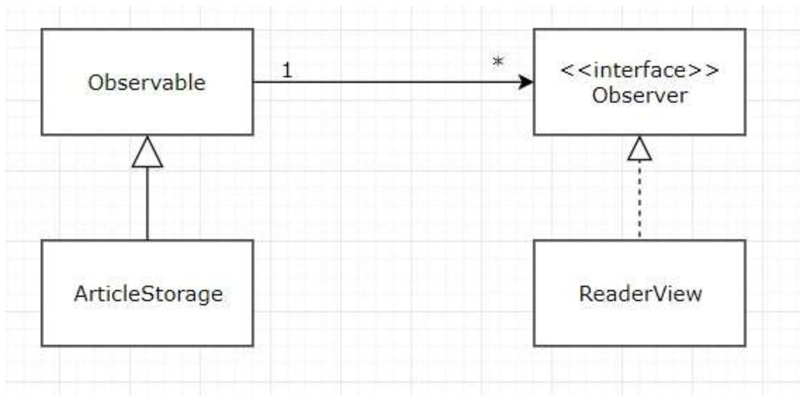
I the case of this application the observers ar the clients which must be notified whenever a change appears in the collection of articles stored by the News Agency. The observable is the collection of articles that changes when an article is added, removed or updated.

How is the Observer Design Pattern implemented:

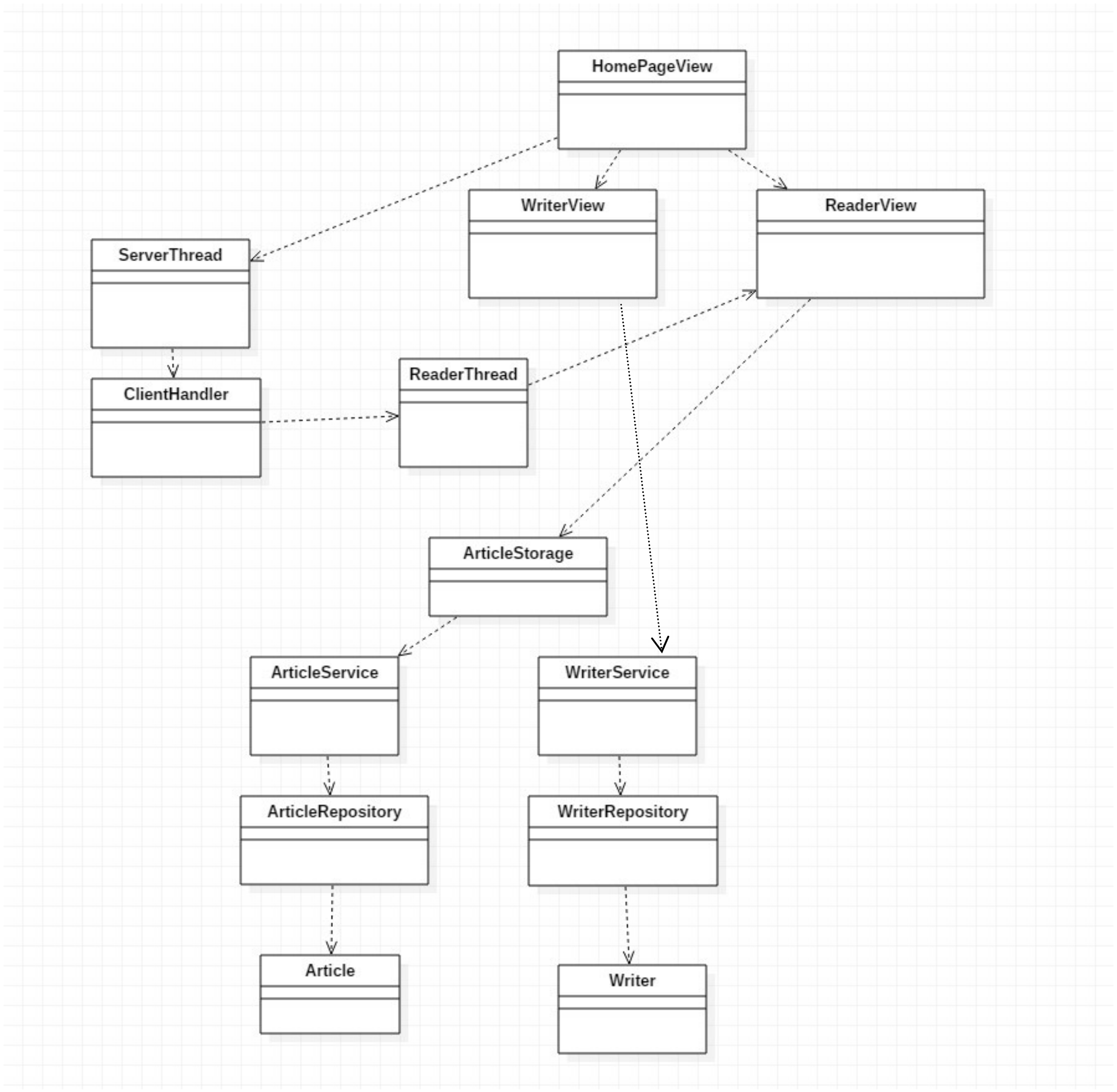
- The class ReaderView implements the Observer interface.
- The class ArticleStorage extends the Observable superclass.
- The class ArticleStorage keeps a collection of the Observers that register to be notifyed when changes appear.
- The class ArticleStorage uses the method notifyObservers() to signal to all

observers that that a change occurred. (an article was added, removed or updtated).

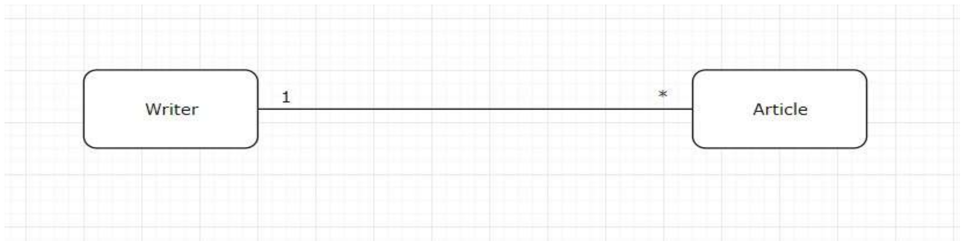
- The ReaderView classes that correspond to each reader that concurrently accesses the application has a method update() which tells the class to update the content of the articles in the list shown to the reader.



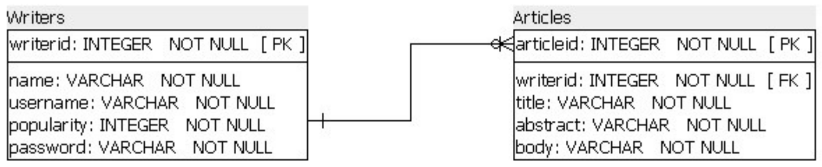
5.2 UML Class Diagram



6. Data Model



I have chosen to represent the data model used in the system`s implementation using the Relational Database Model. In this way both the data, and the exact relationships between it are clearly specified. The database will be created using MariaDB database and the HeidiSQL database management system.


Writers
writerid: INTEGER NOT NULL [PK]
name: VARCHAR NOT NULL
username: VARCHAR NOT NULL
popularity: INTEGER NOT NULL
password: VARCHAR NOT NULL
Articles
articleid: INTEGER NOT NULL [PK]
writerid: INTEGER NOT NULL [FK]
title: VARCHAR NOT NULL
abstract: VARCHAR NOT NULL
body: VARCHAR NOT NULL

7. System Testing

The used testing strategy is Unit Testing:

<u>Unit Testing</u>

By Unit Testing the developer writes a piece of code that executes a specific functionality in the code to be tested ans asserts a certain behavior or state. I choose to use Unit Testing to test methods that implement the CRUD operations on articles and writer accounts.

Individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. When we combine different units that were separately tested (unit testing) we test if the units can
be correctly combined in order to obtain the entire architecture of the system.

I perform Junit testing by testing some of the methods of classes from the Business Layer – Services classes. We create the test classes ArticleServiceTest and WriterServiceTest in order to test some of the methods these classes provide. We use assertions to check if the provided result after is the expected one.

8. Bibliography

https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/
http://www.tutorialspoint.com/json/json_java_example.htm
https://docs.oracle.com/javaee/7/api/javax/json/JsonObject.html
https://www.journaldev.com/1739/observer-design-pattern-in-java