News Agency

Student: Ieremiaș Viorel

**Group: 30432**

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 4

3. System Architectural Design 5

4. UML Sequence Diagrams 7

5. Class Design 7

6. Data Model 8

7. System Testing 8

8. Bibliography 8

1. Requirements Analysis

# Assignment Specification

The following program is a client-server application for a news agency. 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. As a consequence, 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 supports 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.

Other requirements include:

* The application is client-server
* Observer design pattern is used for updating the list of articles in real time
* JSON serialization is used for sending data from the client to the server use
* When writing an article, a list that supports multi-select is shown, for choosing the related articles

# Functional Requirements

Readers have access to the following functions:

* See a list of public articles
* Open an article, with the complete information
* See articles belonging to a writer
* See the related articles of a specific one

The readers are not required and do not have the possibility to log into the system. Their view is accessible without providing any credentials.

By contrast, writers are users that can alter articles, so they are required to provide account information when using these functionalities.

The additional functions a writer has access to are:

* Create a new article
* View
* Edit any section of an article for which he/she is registered as author
* Delete an article

# Non-functional Requirements

* Accessibility – the system must provide a visual interface that will provide easy access to all the functions
* Availability – the system must be operational 24/7, except in cases when planned and announced maintenance procedures are underway
* Capacity – the system must accommodate around 1000 users and must serve 10000 requests per month in the agreed performance parameters
* Extensibility – the system must easily accept extensions, like allowing setting right for different users, ranging from editing, to collaborating or viewing only
* Performance – the system must have a response time of at most 1s
* Platform compatibility – the system must be compatible will all platforms that have installed a Java runtime environment, version 6 or higher
* Safety – the system must provide only a simple, one-step authentication, but constraints must be enforced on the strength of the passwords
* Scalability – the system will be able to accommodate another 1000 users with at most 10% loss of performance; this criterion applies up to a number of 10000 users

2. Use-Case Model

***Use case***: Read article

***Level***: user-goal

***Primary actor***: unregistered user (reader)

***Brief description***: The user opens the interface of the system and the application displays a page of 6 panels containing the title and the abstract of the article. The reader then clicks on one of the panels, to open the full content of the article

***Main success scenario***:

• The user opens the Technix Media application

• The system presents a dashboard containing brief information about a set of articles

• The reader selects an article by clicking a panel

* The application presents the full content of the article

***Extensions***: The application fails to retrieve the data from the server, in which case a page displays an error message.

***Use case***: Delete article

***Level***: user-goal

***Primary actor***: writer

***Brief description***: The writer logs into the system with the predefined account set up by the system administration. A list of articles having the logged-in user as main author is presented. The writer selects the article on which the action is applied. Full info about the article and the available actions are presented. *Delete* button is pressed and the confirmation of the deletion is displayed.

***Main success scenario***:

• Login as writer

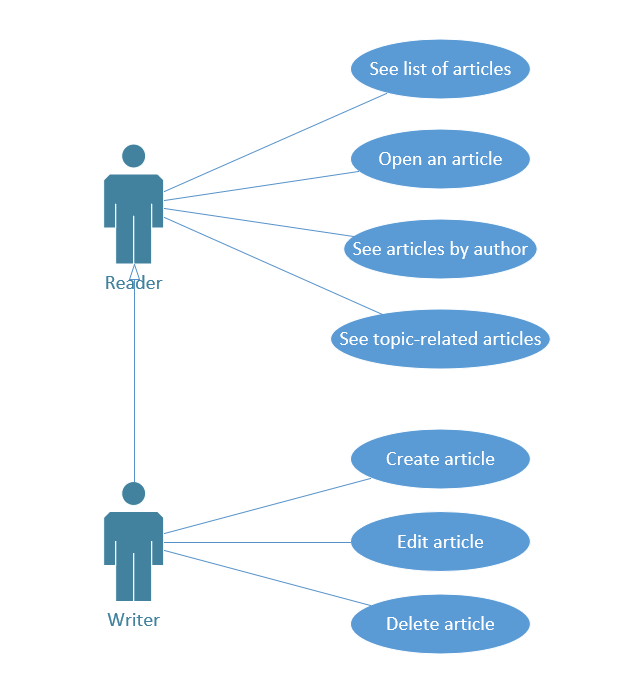
• The system presents a list of the author’s articles

• Writer selects article to read

• The system presents available actions for the article

* The writer selects *Delete* action
* Confirmation of the deletion is displayed

***Extensions***: The writer might fail to login or the system might crash when while accessing the server.



*Use-Case Diagram*

3. System Architectural Design

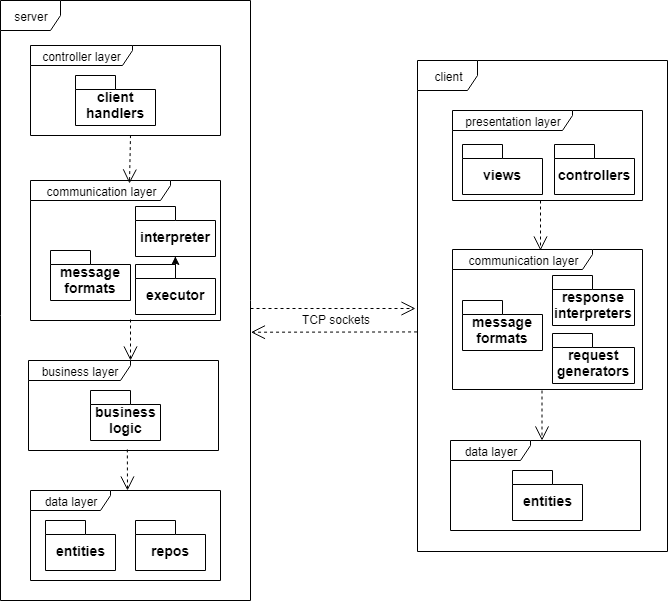
**3.1 Architectural Pattern Description**

The application is architectured as a client-server console application that can serve multiple users at the same time. Both the server and the client are structured in layers, focused on specific tasks: data fetching and persistence management, business processing, input and output handling and communication protocol between the client and the server.

The two components communicate via TCP sockets – point-to-point communication channels, provided also in the Java language, in the java.net package.

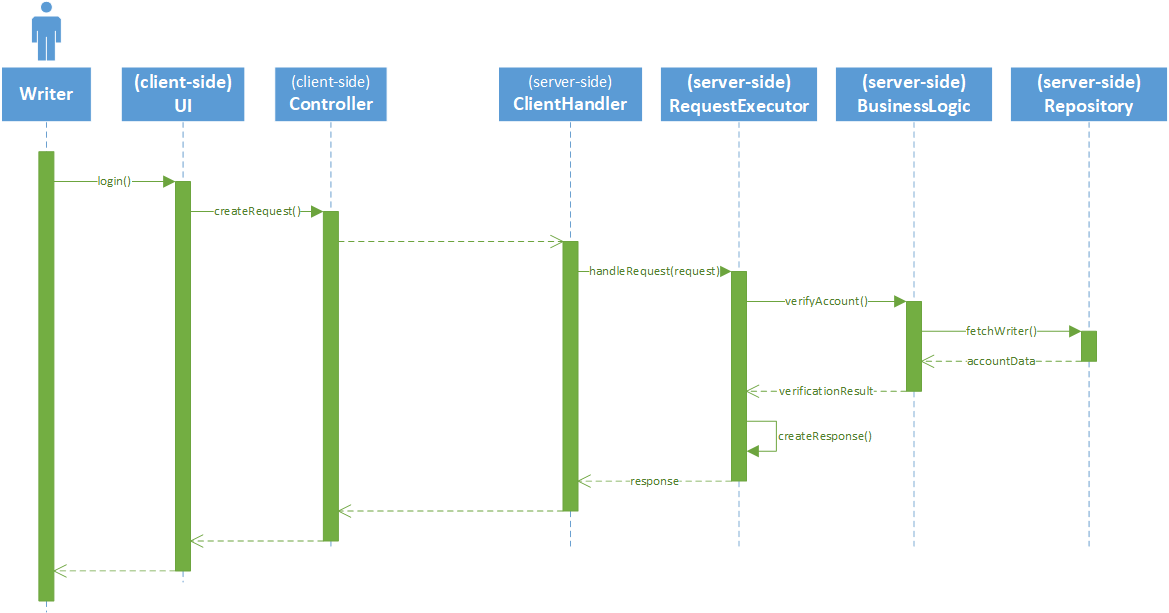
The client and the server exchange messages in the form of commands that contain an identifier and optionally arguments. The commands differentiate themselves as requests and responses, the flow of request being from the client towards the server and the flow of responses being from the server towards the client. Thus, the main components that take care of the message handling are the RequestExecutor in the Server, respectively the ResponseInterpreter in the Client.

**3.2 Diagrams**

****

*Architecture Diagram*

4. UML Sequence Diagrams



*Login Use-Case Sequence Diagram*

5. Class Design

**5.1 Design Patterns Description**

**Command pattern** is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. This information typically includes the method name, the object that owns the method and values for the method parameters.

In the News Agency Application, command pattern is applied with the purpose of establishing a message format protocol for the communication between the server and the client. The communication is most of the time simple, the two entities exchanging simple commands, but in some cases the commands (either requests or responses), require arguments that must be sent in the same time. Command pattern offers a clean way of encapsulating the command type and the command arguments in the same object. More than that, it offers a uniform and change-independent manner of handling commands.

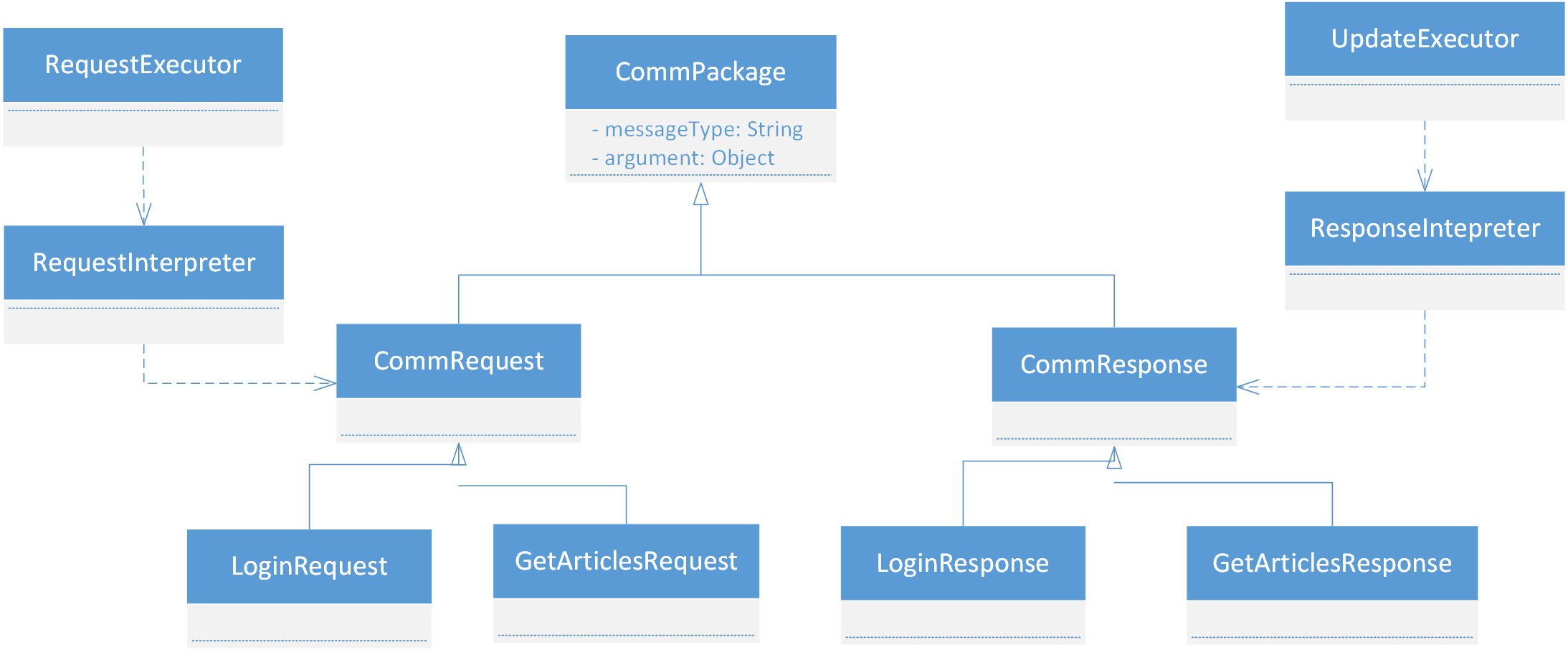
**Strategy pattern** is a behavioral design pattern that enables selecting the algorithm to execute at runtime. Instead of integrating the decision inside the algorithm itself, code receives at runtime instructions as to which algorithm to run from a family of related approaches.

In the News Agency Application, the two programs communicate one with the other by exchanging messages over sockets. Both the client and the server need to choose an action to perform when a request or a response is received, with regards to the type of the massage received, so an approach based on the strategy pattern is best suited.

**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. It is mainly used to implement distributed event handling systems, in "event driven" software.

The application of the observer pattern in the News Agency Application stands in the automatic update of the reader’s dashboard when a change of an article is committed by a writer.

**5.2 UML Class Diagram**



*Communication layer class diagram*

6. Data Model

7. System Testing

8. Bibliography