Assignment 3
Analysis and Design Document

Student: Han Ana-Maria
Group: 30433

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The objective of this assignment is to allow students to become familiar with the client server architectural style and the Observer design pattern.

Use Java/C# API to design and implement a client-server application for a news agency. The application has three types of users: the readers, the writers and an administrator. 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 to create, update or delete articles. The **admin** is the only one who can create writer accounts, but cannot create new admin accounts. So the admin accounts are preset by the application developer and cannot be altered. An article has the following components:

- Title
- Abstract
- Author
- Body
- List of related articles

When reading an article, the user should be able to see the title and the abstract of the related articles. By clicking on the title of the related article, he will be taken to a page that displays the full article. 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

For the administrator:

- Login
- View articles, writers
- Add, delete writers
- Add, delete articles

For the writer:

- Login
- View their own articles
- Add articles

For the reader:

- View articles
- Open article
- View related articles to article currently opened
- No need to login

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.

Client-server architecture

Observer design pattern for updating the list of articles in real time

JSON serialization for sending data from the client to the server

1.3 Non-functional Requirements

- Usability: desktop application, works on computers running Java

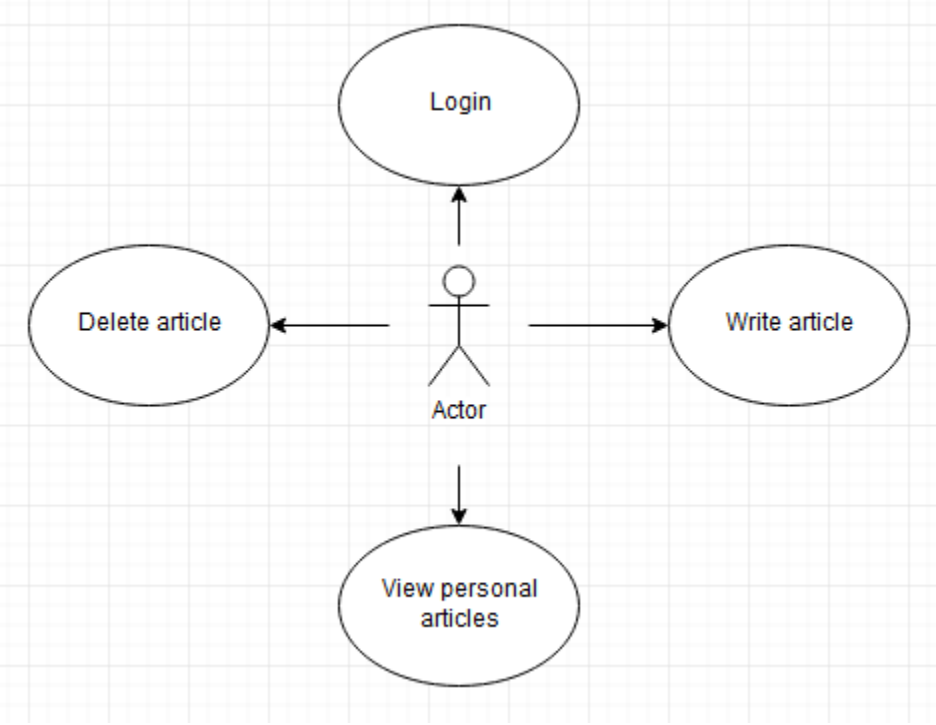
- Performance: a response time of maximum 1 second
- Availability: 20 hours/day, 365 days/year
- Security: password encryption in the database

2. Use-Case Model

Use-case description:

- Use case: write article
- Level: user-goal level
- Primary actor: writer (regular user)
- Main success scenario: the writer logs in successfully, writes the title, abstract and body(content) of the article, clicks on “Load” if they want to load an image(optional), and then clicks on “Submit article” to create it.
- Extensions:
    - Writer doesn’t enter the right credentials and cannot login
    - Writer clicks on “Submit article” without previously inserting data in all the fields

Use-case diagram for writer



:

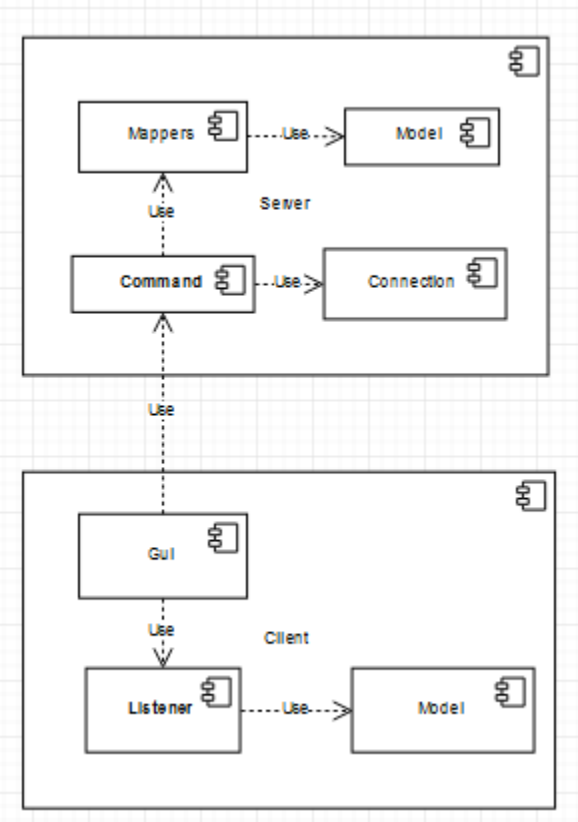
3. System Architectural Design

3.1 Architectural Pattern Description

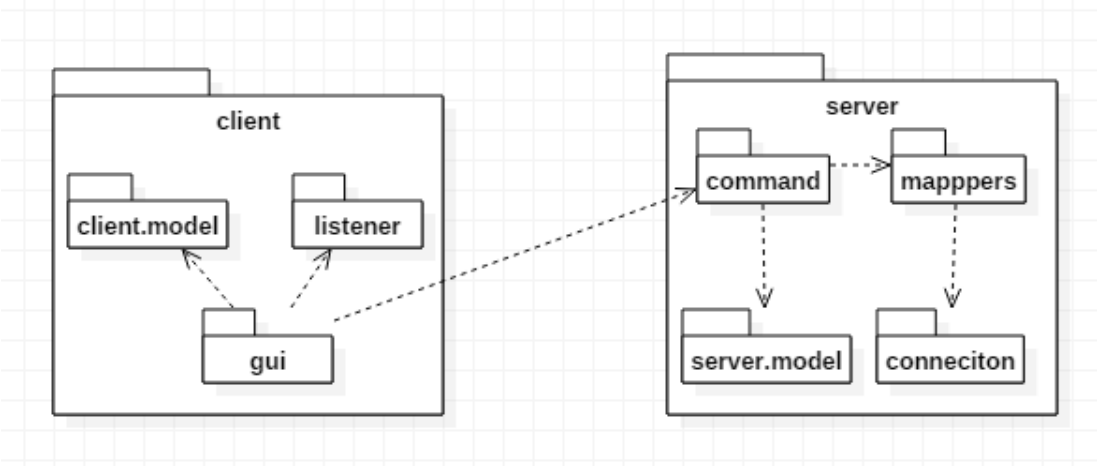
The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. [1]

3.2 Diagrams

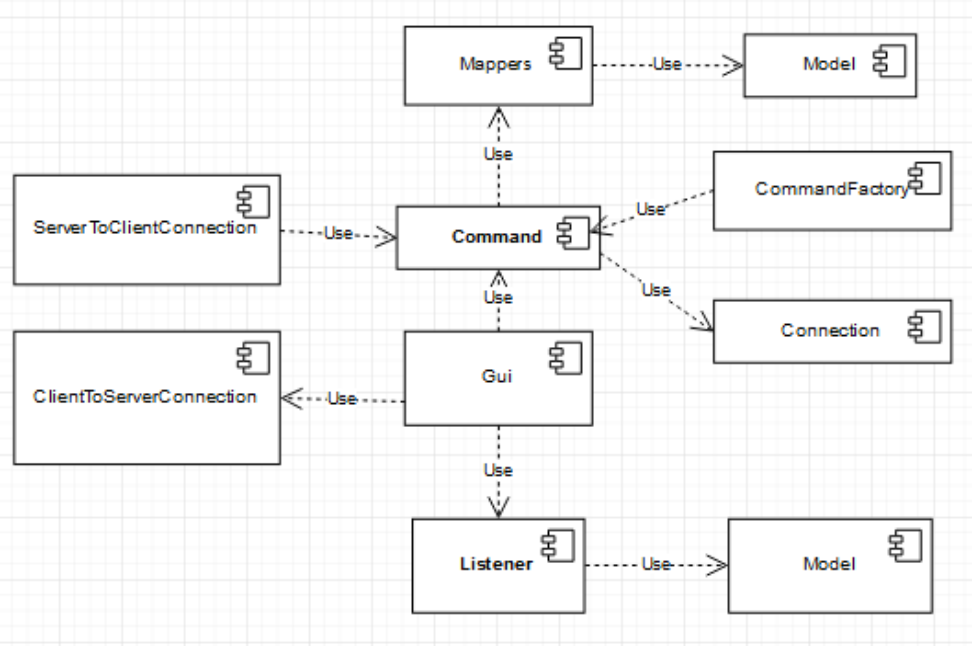
Deployment diagram:



Package diagram:

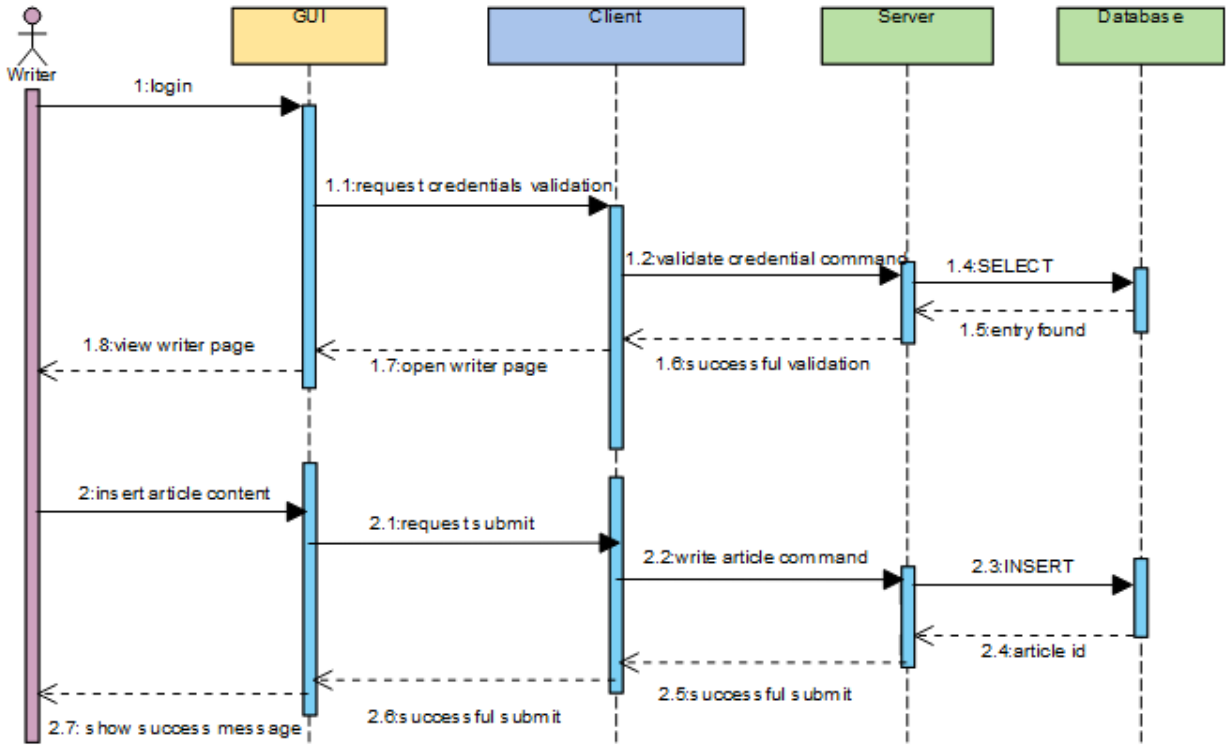


Component diagram:



4. UML Sequence Diagrams

Sequence diagram for writing an article:



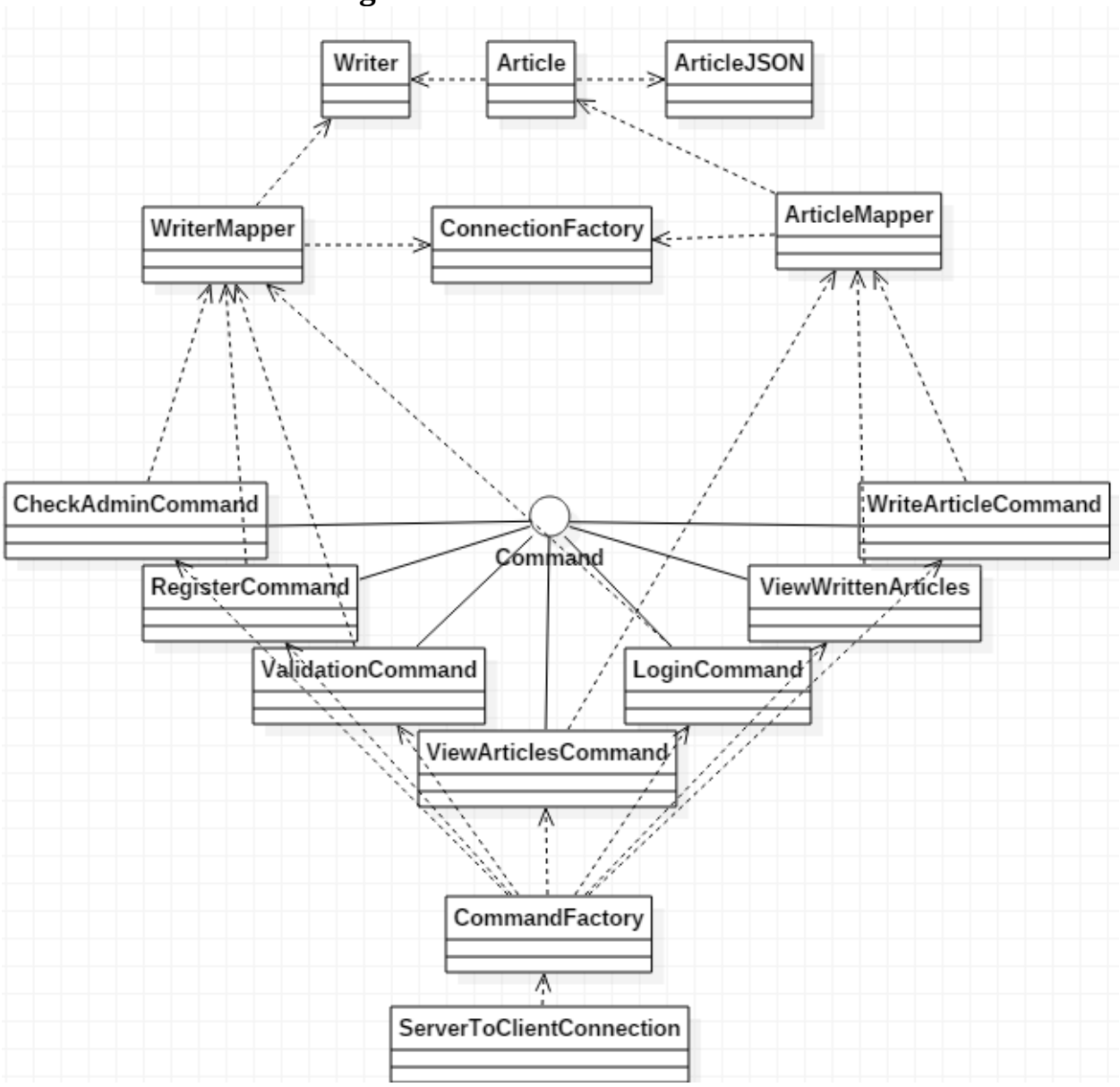
5. Class Design

5.1 Design Patterns Description

5.1.1 Command Design Pattern

It 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 includes the method name, the object that owns the method and values for the method parameters. [2] A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

5.2 UML Class Diagram



6. Data Model

The data model consists in the following classes:

- Writer: because we need to know the credentials of those who are able to write articles for the agency. A writer has a name, username, password, and a list of articles they have written so far, which can change (the writer can add or delete their own articles).
- ArticleJSON: how the content of an article is stored in a .JSON file. It contains the title, the abstract, the body, and (optionally) an image – the path for an image, actually.
- Article: consists in a title, abstract, list of related articles, location of the corresponding .JSON file on disk and an ArticleJSON object.

7. System Testing

8. Bibliography

[1] Client-server model: https://en.wikipedia.org/wiki/Client%E2%80%93server_model
[2] Command design pattern: https://en.wikipedia.org/wiki/Command_pattern
[3] Useful tutorials: https://github.com/buzea/SoftwareDesign2018