# C++ Programming Project

EDAF50

*Alexander Magnusson - al0173ma-s@student.lu.se*

*Max Soller - ma8251so-s@student.lu.se*

*Fredrik Horn Dannert - fr3632ho-s@student.lu.se*

May 9, 2022

# 1 Introduction - Project Description

In this assignment the group has been tasked with implementing a news system, written in C++. The news system shall consist of a news- server and client. The servers will be dedicated, hence, they will not be communicating with other servers.

Moreover, the server was to be implemented in two separate versions. One, which is volatile and another which is persistent. Persistent meaning it is stored on disk and volatile implying storage is stored temporarily in memory. Lastly, a client was devised, allowing for server interaction through the terminal.

See Appendix for a UML-diagram and a Sequence-diagram of the system.

# 2 Classes and Responsibilities

## 2.1 Server

The server, **myserver.cc**, has the responsibility to listen for requests from the client and returning values accordingly. The server and client communicate with a connection on a port on localhost and follows a predetermined protocol, which follows the following structure: **<commandbyte> <parameters> <endbyte>**. When the server has been given a request it uses the class **MessageProtocol** to retrieve, delete or add data to the database.

## 2.2 Client

The user connects to the database using the client, **myclient.cc**. It sends different requests to list, retrieve or create newsgroups and articles. It connects to the server using a localhost port.

## 2.3 Message Protocol

**MessageProtocol** takes care of all forms of message handling between the client and server. In the specifications of the task we were given a structure to follow when reading strings and numbers as input to our program as well as dealing with output from our program. Requests such as creating news groups and articles, fetching said groups and articles is processed by the MessageProtocol class. A client's request will be, when interacting with the server, indirectly processed by the MessageProtocol class. MessageProtocol will ensure that the right messages are written to the log through the connection class and also ensure that the client gets the correct information.

## 2.4 Database

The Database is where all the data is stored. The system has two different kinds of databases, one that stores the data in primary memory, **VolatileDatabase**, and one that stores the data on the disk, *DiskDatabase.*

#### 2.4.1 Volatile Database

The Volatile Database stores the data in primary memory using data structures from the c++ standard library. When the session ends the data will therefore be lost.

#### 2.4.2 Disk Database

The Disk Database will instead store the data on the disk using a file structure. With this data can be stored between sessions. The database is in practice a directory where it has sub directories for every News group and every article is a *txt* file in the respective news group directory. The name of each news group directory and article *txt* file is their id. To be able to preserve the names of the news group there is also a file called *id_newsgroups_map.txt* which holds the id and name for each news group. When the session is started the disk database collects the newsgroups from this file so it knows which newsgroups existed from previous sessions.

### 2.5 Newsgroup

This class is responsible for representing a certain news group in the system. It has an id, a name and a map which stores the articles in this news group as values and the article id as keys.

### 2.6 Article

This class is responsible for representing a article in the system. It has an id and three string attributes for title, author and text.

## 3 Conclusion

The program fulfills its criterion, however, there is of course room for improvement.

# 4 Appendix

## 4.1 UML - Diagram

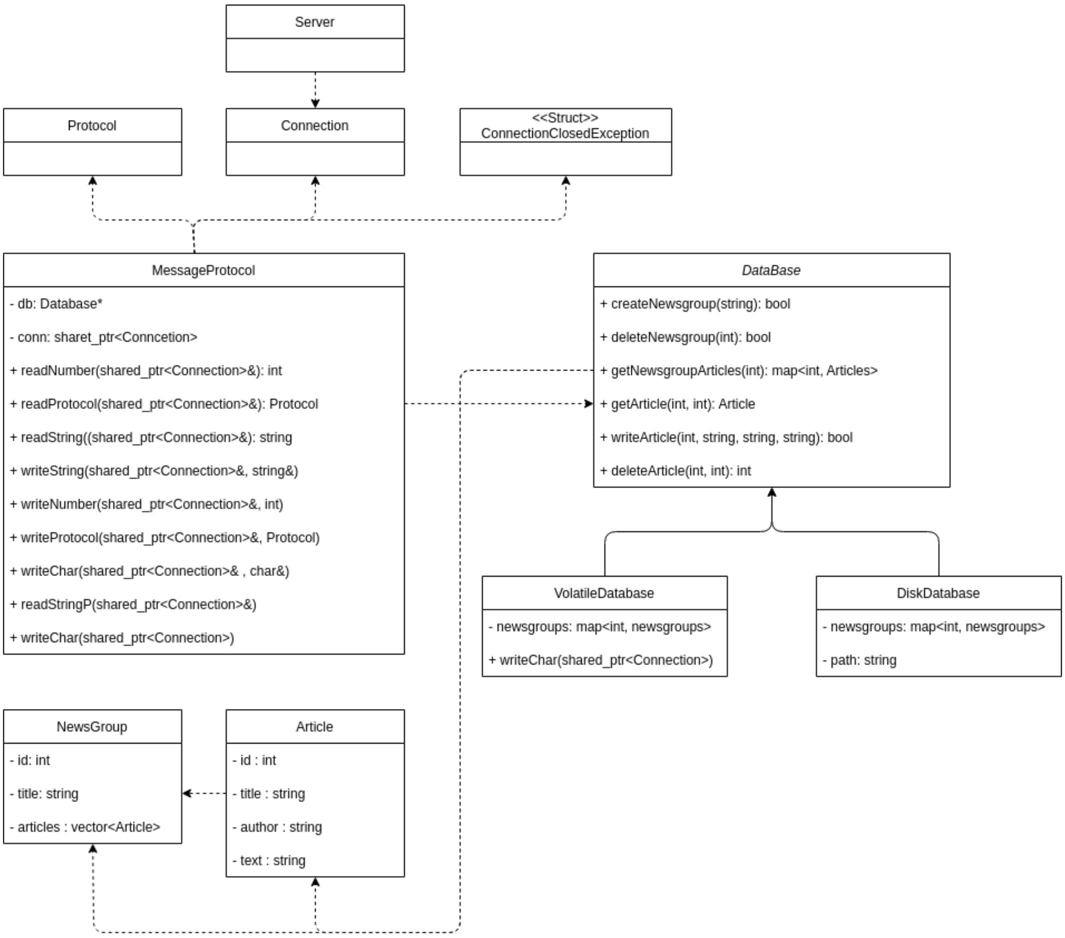
Server
Protocol
Connection
<<Struct>>
ConnectionClosedException
MessageProtocol
- db: Database*
- conn: sharet_ptr<Conncetion>
+ readNumber(shared_ptr<Connection>&): int
+ readProtocol(shared_ptr<Connection>&): Protocol
+ readString((shared_ptr<Connection>&): string
+ writeString(shared_ptr<Connection>&, string&)
+ writeNumber(shared_ptr<Connection>&, int)
+ writeProtocol(shared_ptr<Connection>&, Protocol)
+ writeChar(shared_ptr<Connection>& , char&)
+ readStringP(shared_ptr<Connection>&)
+ writeChar(shared_ptr<Connection>)
DataBase
+ createNewsgroup(string): bool
+ deleteNewsgroup(int): bool
+ getNewsgroupArticles(int): map<int, Articles>
+ getArticle(int, int): Article
+ writeArticle(int, string, string, string): bool
+ deleteArticle(int, int): int
VolatileDatabase
- newsgroups: map<int, newsgroups>
+ writeChar(shared_ptr<Connection>)
DiskDatabase
- newsgroups: map<int, newsgroups>
- path: string
NewsGroup
- id: int
- title: string
- articles : vector<Article>
Article
- id : int
- title : string
- author : string
- text : string

Figure 1: UML diagram of the system

### 4.2 Sequence Diagram

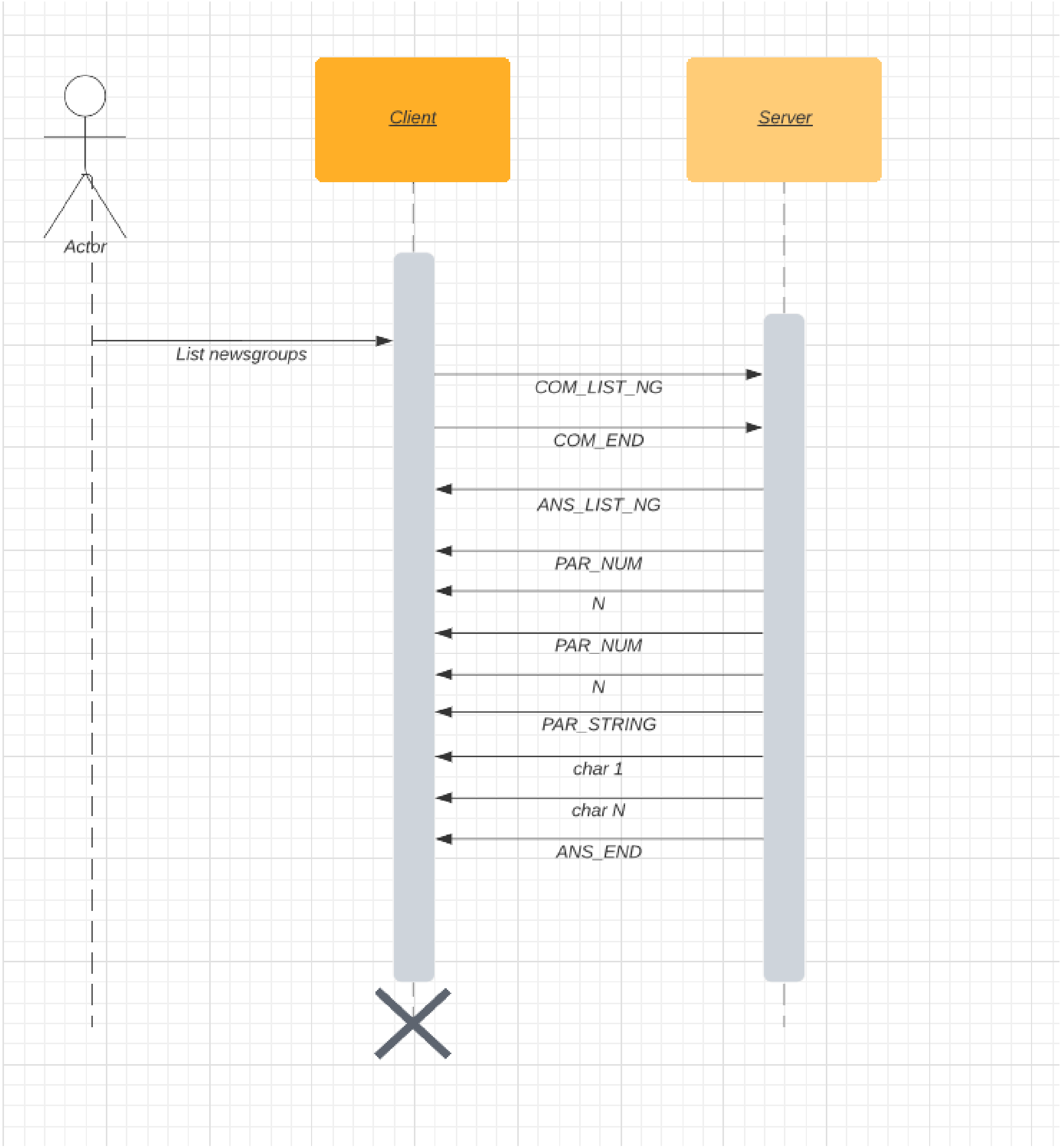

Figure 2: UML diagram of the system