Offline Messenger Documentation
Computer Networks

Rusu Ioana, 3E3

Faculty of Computer Science, Iasi

1 Introduction

Offline Messenger is a client/server application designed to facilitate message exchange between connected users, including the ability to send messages to offline users. This platform ensures that users receive messages when they reconnect to the server, letting them know through a notification. Additionally, users have the capability to respond specifically to certain received messages and the functionality to track conversation history for each individual user.

2 Used techonologies

A. Protocol

I chose to use TCP, Transmission Control Protocol, in favor of UDP, User Datagram Protocol, because of its reliability in maintaining a stable connection between the client and the server. UDP prioritizes speed in transmitting information, while TCP ensures that all information is successfully received, even if it takes a longer period of time. Since the project involves the exchange of messages, it is crucial to prevent information loss. This reliability is guaranteed by the Transmission Control Protocol.

B. Threads

Given that the primary objective of my project is to enable users to exchange multiple messages simultaneously, I opted to implement the server using multiple threads. This ensures the server's concurrency, allowing users to communicate with each other in parallel without waiting for one another.

C. Database

I have chosen SQLite3 as the database for my project. The database will consist of a minimum of two tables: *utilizatori*, containing user information like *nume*, *parola* and *notificari*, storing information about the sender and receiver of offline messages. Additionally, when a conversation is created, a new table will be dynamically generated, joining the names of the users involved. This table will store the message's ID, the sender of the message, and the message text. To facilitate replies, the text to which the reply is made will be enclosed within parentheses in the respective table.

3 Application Structure

The clients are going to be able to use multiple commands:

– *login* ⟨*nume-utilizator*⟩ ⟨*parola*⟩: the user can get into their account using their username and password;
– *help*: displays the available commands for an user give his status (logged in or not);
– *logout*: after logging in the user is able to log out
– *contacte*: the user can see a list of all the users available for conversations
– *mesaje* ⟨*nume-contact*⟩: the user is able to view the past messages exchanged between him and the user inputed;
– *sterge* ⟨*nume-contact*⟩: the user can delete the conversation between him and the user inputed;
– *scrie* ⟨*nume-contact*⟩ ⟨*mesaj*⟩: the user can send massage to another user;
– *raspunde* ⟨*nume-contact*⟩ ⟨*id-mesaj*⟩ ⟨*mesaj*⟩: the user can reply to a certain message in a conversation;

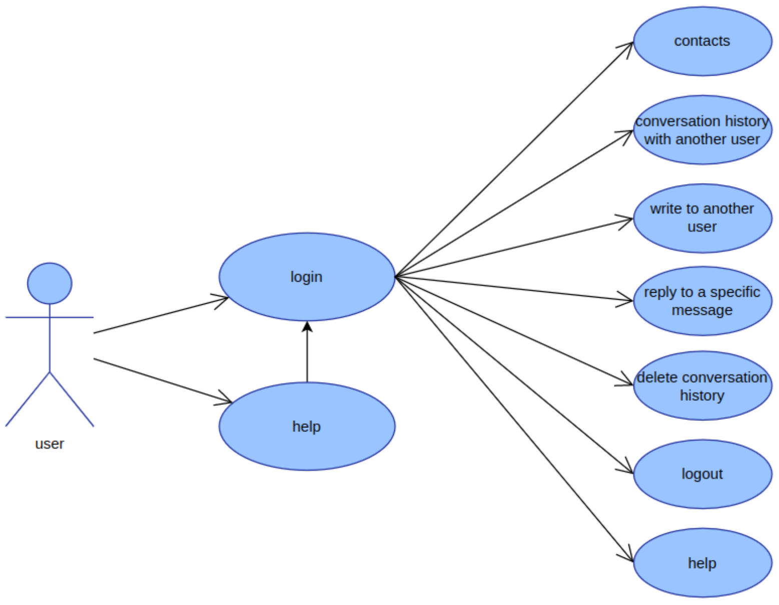

contacts
conversation history with another user
write to another user
reply to a specific message
delete conversation history
logout
help
login
help
user

Fig. 1. Use-case diagram

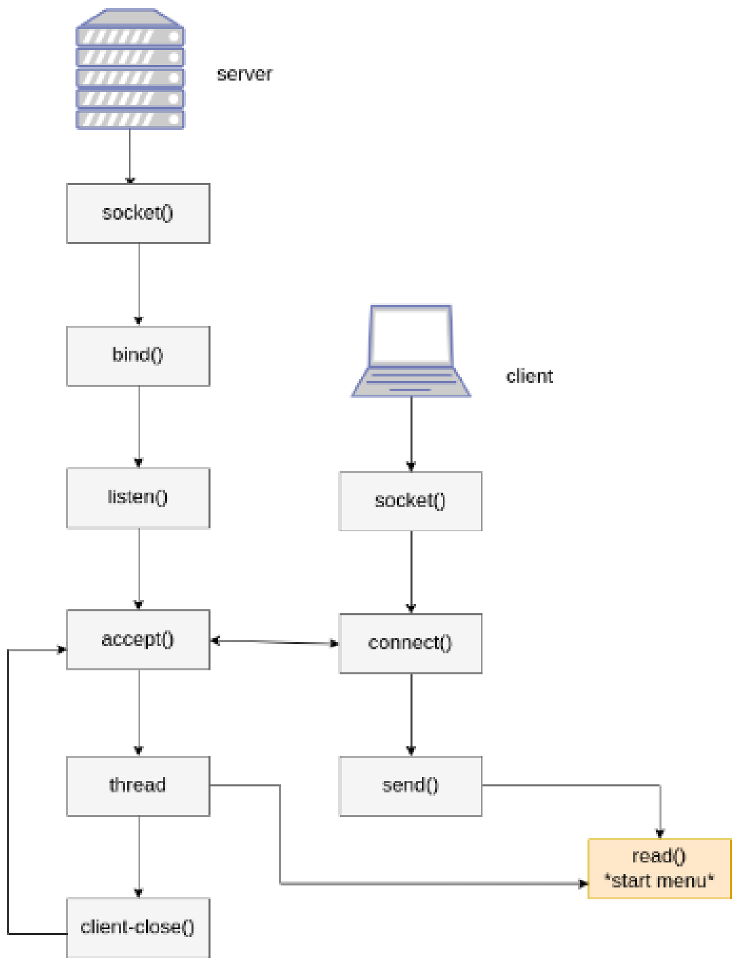


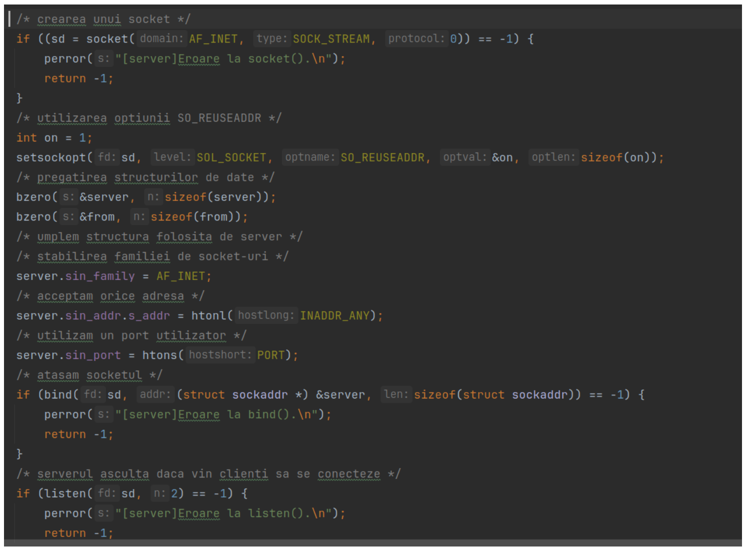
Fig. 2. Connection between client and server

4 Implementation Aspects

1. Code

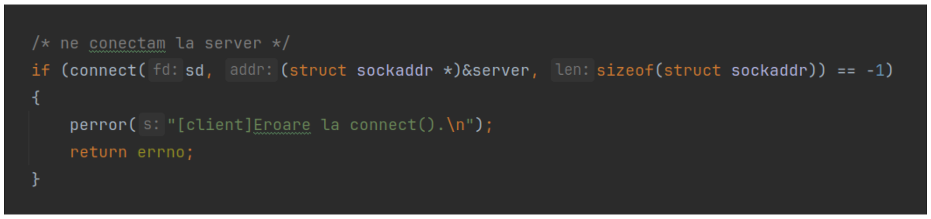
The communication in the project ”Offline Messenger” is a client/server type using sockets. Multiple clients are going to be connected on the server at the same time.

The server initiates a socket upon startup and enters a listening state, awaiting connections from users who will transmit information through the client. Upon successful connections, the server spawns a dedicated thread for each client.



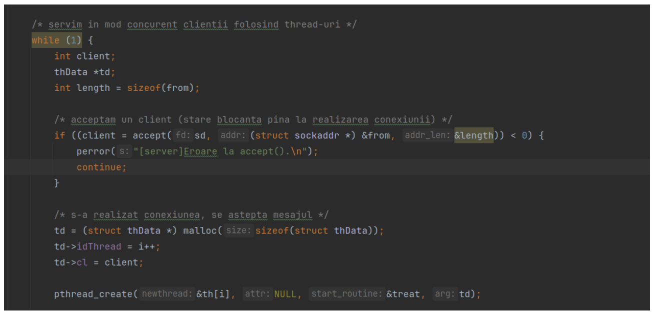
```
/* crearea unui socket */
if ((sd = socket( domain: AF_INET,  type: SOCK_STREAM,  protocol: 0)) == -1) {
    perror( s: "[server]Eroare la socket().\n");
    return -1;
}
/* utilizarea optiunii SO_REUSEADDR */
int on = 1;
setsockopt( fd: sd,  level: SOL_SOCKET,  optname: SO_REUSEADDR,  optval: &on,  optlen: sizeof(on));
/* pregatirea structurilor de date */
bzero( s: &server,  n: sizeof(server));
bzero( s: &from,  n: sizeof(from));
/* umplem structura folosita de server */
/* stabilirea familiei de socket-uri */
server.sin_family = AF_INET;
/* acceptam orice adresa */
server.sin_addr.s_addr = htonl( hostlong: INADDR_ANY);
/* utilizam un port utilizator */
server.sin_port = htons( hostshort: PORT);
/* atasam socketul */
if (bind( fd: sd,  addr: (struct sockaddr *) &server,  len: sizeof(struct sockaddr)) == -1) {
    perror( s: "[server]Eroare la bind().\n");
    return -1;
}
/* serverul asculta daca vin clienti sa se conecteze */
if (listen( fd: sd,  n: 2) == -1) {
    perror( s: "[server]Eroare la listen().\n");
    return -1;
```

Fig. 3. Socket use in server



```
/* ne conectam la server */
if (connect( fd: sd,  addr: (struct sockaddr *)&server,  len: sizeof(struct sockaddr)) == -1)
{
    perror( s: "[client]Eroare la connect().\n");
    return errno;
}
```

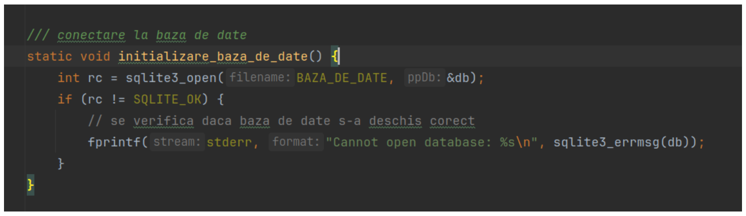
Fig. 4. Client connection



```
/* servim in mod concurent clientii folosind thread-uri */
while (1) {
    int client;
    thData *td;
    int length = sizeof(from);

    /* acceptam un client (stare blocanta pina la realizarea conexiunii) */
    if ((client = accept( fd: sd,  addr: (struct sockaddr *) &from,  addr_len: &length)) < 0) {
        perror( s: "[server]Eroare la accept().\n");
        continue;
    }

    /* s-a realizat conexiunea, se astepta mesajul */
    td = (struct thData *) malloc( size: sizeof(struct thData));
    td->idThread = i++;
    td->cl = client;

    pthread_create( newthread: &th[i],  attr: NULL,  start_routine: &treat,  arg: td);
```

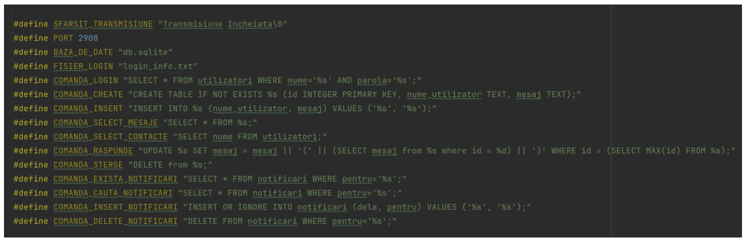
Fig. 5. Thread creation

In each function used for implementing the commands there will be used pre-defined sql queries.



```
/// conectare la baza de date
static void initializare_baza_de_date() {
    int rc = sqlite3_open( filename: BAZA_DE_DATE,  ppDb: &db);
    if (rc != SQLITE_OK) {
        // se verifica daca baza de date s-a deschis corect
        fprintf( stream: stderr,  format: "Cannot open database: %s\n", sqlite3_errmsg(db));
    }
}
```

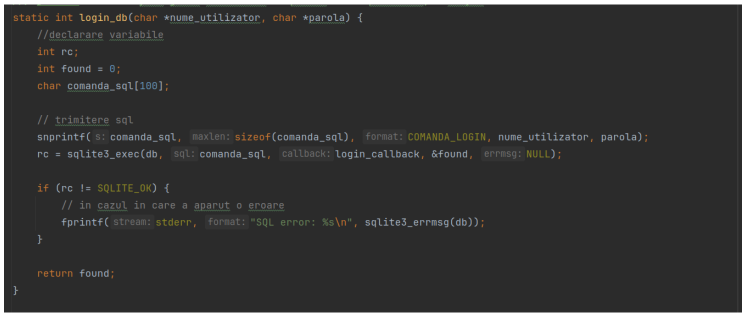
Fig. 6. Database connection



```
#define SFARSIT_TRANSMISIUNE "Transmisiune Incheiata\0"
#define PORT 2908
#define BAZA_DE_DATE "db.sqlite"
#define FISIER_LOGIN "login_info.txt"
#define COMANDA_LOGIN "SELECT * FROM utilizatori WHERE nume='%s' AND parola='%s';"
#define COMANDA_CREATE "CREATE TABLE IF NOT EXISTS %s (id INTEGER PRIMARY KEY, nume_utilizator TEXT, mesaj TEXT);"
#define COMANDA_INSERT "INSERT INTO %s (nume_utilizator, mesaj) VALUES ('%s', '%s');"
#define COMANDA_SELECT_MESAJE "SELECT * FROM %s;"
#define COMANDA_SELECT_CONTACTE "SELECT nume FROM utilizatori;"
#define COMANDA_RASPUNDE "UPDATE %s SET mesaj = mesaj || '(' || (SELECT mesaj from %s where id = %d) || ')' WHERE id = (SELECT MAX(id) FROM %s);"
#define COMANDA_STERGE "DELETE from %s;"
#define COMANDA_EXISTA_NOTIFICARI "SELECT * FROM notificari WHERE pentru='%s';"
#define COMANDA_CAUTA_NOTIFICARI "SELECT * FROM notificari WHERE pentru='%s';"
#define COMANDA_INSERT_NOTIFICARI "INSERT OR IGNORE INTO notificari (dela, pentru) VALUES ('%s', '%s');"
#define COMANDA_DELETE_NOTIFICARI "DELETE FROM notificari WHERE pentru='%s';"
```

Fig. 7. Queries

In order to use the application, the user must be logged in.

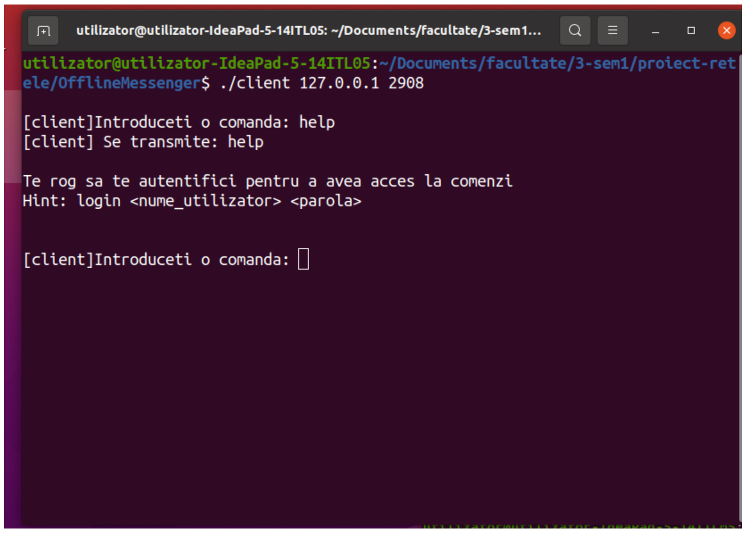


```c
static int login_db(char *nume_utilizator, char *parola) {
    //declarare variabile
    int rc;
    int found = 0;
    char comanda_sql[100];

    // trimitere sql
    snprintf( s: comanda_sql,  maxlen: sizeof(comanda_sql),  format: COMANDA_LOGIN, nume_utilizator, parola);
    rc = sqlite3_exec(db,  sql: comanda_sql,  callback: login_callback, &found,  errmsg: NULL);

    if (rc != SQLITE_OK) {
        // in cazul in care a aparut o eroare
        fprintf( stream: stderr,  format: "SQL error: %s\n", sqlite3_errmsg(db));
    }

    return found;
}
```

Fig. 8. Login

2. Real usage scenarios

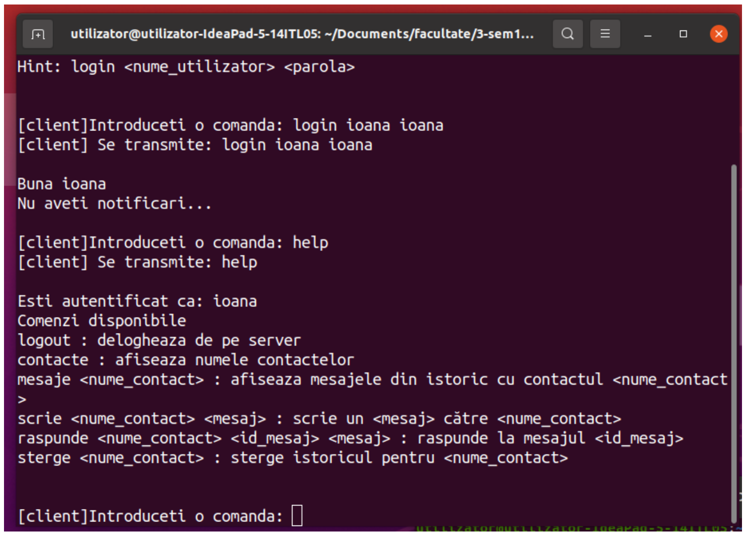
In this part of the documentation, I will present usage scenerios including the process of logging in and the commands available for users.



utilizator@utilizator-IdeaPad-5-14ITL05: ~/Documents/facultate/3-sem1...

```
utilizator@utilizator-IdeaPad-5-14ITL05:~/Documents/facultate/3-sem1/proiect-ret
ele/OfflineMessenger$ ./client 127.0.0.1 2908

[client]Introduceti o comanda: help
[client] Se transmite: help

Te rog sa te autentifici pentru a avea acces la comenzi
Hint: login <nume_utilizator> <parola>


[client]Introduceti o comanda: 
```

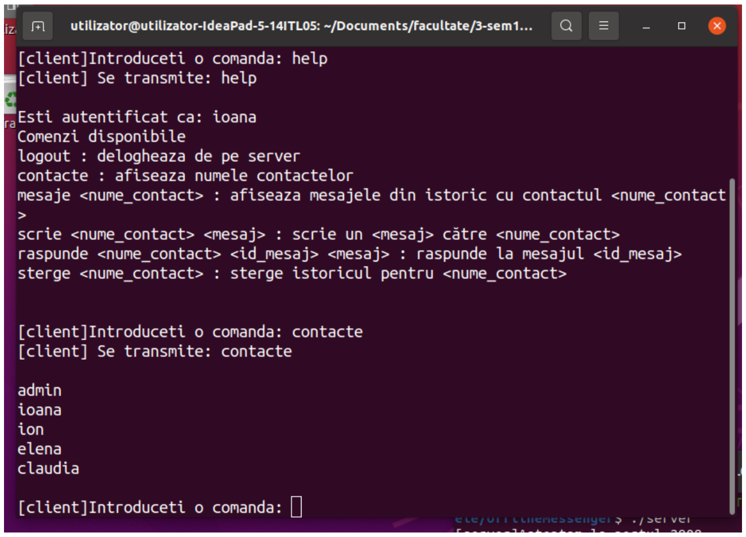
Fig. 9. User not logged in, using the 'help' command for displaying the available commands



utilizator@utilizator-IdeaPad-5-14ITL05: ~/Documents/facultate/3-sem1...

```
Hint: login <nume_utilizator> <parola>


[client]Introduceti o comanda: login ioana ioana
[client] Se transmite: login ioana ioana

Buna ioana
Nu aveti notificari...

[client]Introduceti o comanda: help
[client] Se transmite: help

Esti autentificat ca: ioana
Comenzi disponibile
logout : delogheaza de pe server
contacte : afiseaza numele contactelor
mesaje <nume_contact> : afiseaza mesajele din istoric cu contactul <nume_contact
>
scrie <nume_contact> <mesaj> : scrie un <mesaj> către <nume_contact>
raspunde <nume_contact> <id_mesaj> <mesaj> : raspunde la mesajul <id_mesaj>
sterge <nume_contact> : sterge istoricul pentru <nume_contact>


[client]Introduceti o comanda: 
```

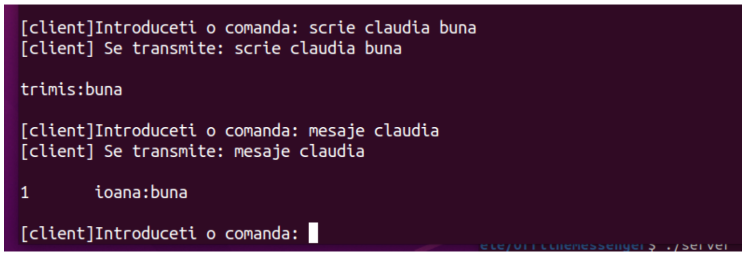
Fig. 10. User logged in, using the 'help' command for displaying the available commands



utilizator@utilizator-IdeaPad-5-14ITL05: ~/Documents/facultate/3-sem1...

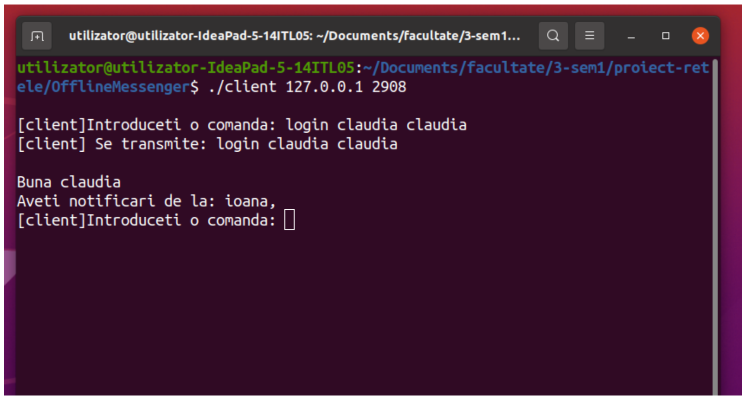
```
[client]Introduceti o comanda: help
[client] Se transmite: help

Esti autentificat ca: ioana
Comenzi disponibile
logout : delogheaza de pe server
contacte : afiseaza numele contactelor
mesaje <nume_contact> : afiseaza mesajele din istoric cu contactul <nume_contact
>
scrie <nume_contact> <mesaj> : scrie un <mesaj> către <nume_contact>
raspunde <nume_contact> <id_mesaj> <mesaj> : raspunde la mesajul <id_mesaj>
sterge <nume_contact> : sterge istoricul pentru <nume_contact>


[client]Introduceti o comanda: contacte
[client] Se transmite: contacte

admin
ioana
ion
elena
claudia

[client]Introduceti o comanda: 
```

Fig. 11. The 'contacte' command showing available users



```
[client]Introduceti o comanda: scrie claudia buna
[client] Se transmite: scrie claudia buna

trimis:buna

[client]Introduceti o comanda: mesaje claudia
[client] Se transmite: mesaje claudia

1       ioana:buna

[client]Introduceti o comanda: 
```

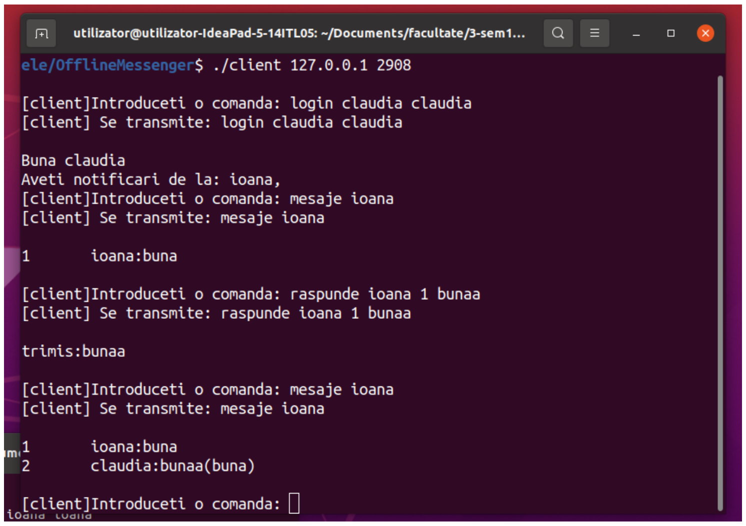
Fig. 12. The ’scrie’ and ’mesaje’ commands displaying the functionality of sending a message and viewing it in history



utilizator@utilizator-IdeaPad-5-14ITL05: ~/Documents/facultate/3-sem1...

```
utilizator@utilizator-IdeaPad-5-14ITL05:~/Documents/facultate/3-sem1/proiect-ret
ele/OfflineMessenger$ ./client 127.0.0.1 2908

[client]Introduceti o comanda: login claudia claudia
[client] Se transmite: login claudia claudia

Buna claudia
Aveti notificari de la: ioana,
[client]Introduceti o comanda: 
```

Fig. 13. Logging in as the receiver user and being notified about the messages received while offline



utilizator@utilizator-IdeaPad-5-14ITL05: ~/Documents/facultate/3-sem1...

```
ele/OfflineMessenger$ ./client 127.0.0.1 2908

[client]Introduceti o comanda: login claudia claudia
[client] Se transmite: login claudia claudia

Buna claudia
Aveti notificari de la: ioana,
[client]Introduceti o comanda: mesaje ioana
[client] Se transmite: mesaje ioana

1       ioana:buna

[client]Introduceti o comanda: raspunde ioana 1 bunaa
[client] Se transmite: raspunde ioana 1 bunaa

trimis:bunaa

[client]Introduceti o comanda: mesaje ioana
[client] Se transmite: mesaje ioana

1       ioana:buna
2       claudia:bunaa(buna)

[client]Introduceti o comanda: 
```

Fig. 14. Replying to a specific and viewing it in history

5 Conclusions

The project's plan that I've presented in this document can definetly be improved by fixing mistakes and adding new features:

- I think it would be important to have a command which allows you to block certain users from sending you messages and be able to change your own status to "don't disturb" or "invisible" to certain users;
- It would be interesting if the users were able to form groupchats.

References

1. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
2. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
3. https://zetcode.com/db/sqlitec/
4. https://beej.us/guide/bgnet/html/multi/index.html
5. https://beej.us/guide/bgnet/html/multi/index.html