REȚELE DE CALCULATOARE
TEMĂ 2
Raport tehnic
Offline Messenger

Radu Robert-Andrei

December 19, 2022

Offline Messenger(B)
Sa se dezvolte o aplicatie client/server care sa permita schimbul de mesaje intre utilizatori care sunt conectati si sa ofere functionalitatea trimiterii mesajelor si catre utilizatorii offline, acestora din urma aparandu-le mesajele atunci cand se vor conecta la server. De asemenea, utilizatorii vor avea posibilitatea de a trimite un raspuns (reply) in mod specific la anumite mesaje primite. Aplicatia va oferi si istoricul conversatiilor pentru si cu fiecare utilizator in parte.

1 Introducere

Proiectul Offline Messenger este o aplicatie de tip client/server prin intermediul careia utilizatorii isi pot trimite mesaje. Acestea au posibilitatea de a trimite mesaje utilizatorilor online cat si celor offline, acestia primind mesajul la momentul conectarii. Exista posibilitatea de a da reply doar la anumite mesaje iar datele despre utilizatori cat si mesajele sunt stocate intr-o baza de date, utilizatorii avand optiunea de a vedea istoricul conversatiilor.

2 Tehnologii utilizate

→ Limbajele folosite pentru implementare au fost C/C++
→ Ca si protocol de comunicare am folosit TCP, fiind un protocol orientat conexiune. Fiind o aplicatie de comunicat pierderea de mesaje este un lucru de nedorit(in loc de UDP care nu este orientat conexiune si exista posibilitatea de pierdere de informatie)
→ Pentru gestionarea proceselor se folosesc threadurile in loc de **Process identifier(PID)** fiind mult mai usor de folosit pentru aplicatiile de genul acesta unde procesele trebuie sa lucreze in paralel.
→ Pentru stocarea datelor precum *numele de utilizator, mesaje, parole, id* am folosit MqSQL. Permite o interogare, modificare si stergere a datelor mult mai usoara decat daca le-am fi stocat intr-un fisier simplu.

3 Arhitectura aplicatiei

Comunicarea dintre server si client se realizeaza prin socket-uri. Pe server se pot conecta in acelasi timp mai mult clienti, server-ul lucrand in paralel pentru fiecare.

- **register**
  Comanda pentru inregistrare.
- **login**
  Comanda pentru a se loga in aplicatie.
- **sendmsgto <name>**
  Comanda pentru a trimite un mesaj unui utilizator ales.

- **users**
  Comanda pentru a vedea utilizatorii(atat online cat si offline).
- **msghistory <name>**
  Comanda pentru a vedea istoricul unei conversatii.
- **reply <name> <messageId>**
  Comanda pentru a da reply la un anumit mesaj.
- **setnameto <name>**
  Comanda pentru a modifica numele.
- **setpassto <name>**
  Comanda pentru a modifica parola.
- **logout**
  Comanda pentru a se deconecta.
- **quit**
  Comanda pentru a inchide aplicatia.

4 Detalii de implementare

Crearea unui socket

```
if ((socketDesc = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    perror("Eroare socket!\n");
```

Verificare baza de date

```
int verifyDatabase(MYSQL *conn, char *db_name)
{
    char query[256];

    sprintf(query, "SHOW DATABASES LIKE '%s'", db_name);

    if (mysql_query(conn, query))
    {
        printf("Err:Checking db: %s\n", mysql_error(conn));
        return 0;
    }

    MYSQL_RES *result = mysql_store_result(conn);

    int num_rows = mysql_num_rows(result);

    return num_rows > 0;
}
```

Comanda Register

```
void Register(int desc, thData th)
{
    printf("Am intrat in register\n");
    fflush(stdout);

    MYSQL *conn = mysql_init(NULL);

    char query[256];
    char username[100];
    char password[100];
    char userData[100];
    char *answer = (char *)malloc(50 * sizeof(char));

    if (read(desc, &userData, sizeof(userData)) < 0)
    {
        perror("Eroare read username\n");
    }

    sscanf(userData, "%s %s", username, password);

    printf("Name: %s\n", username);
    printf("Password: %s\n", password);
    fflush(stdout);
    if (conn == NULL)
    {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1);
    }

    if (!mysql_real_connect(conn, MYSQL_HOST, MYSQL_USER, MYSQL_PASSWORD,
        MYSQL_DATABASE, 0, NULL, 0))
    {
        fprintf(stderr, "%s\n", mysql_error(conn));
        mysql_close(conn);
        exit(2);
    }

    if (verifyUser(username) == 0)
    {
        sprintf(query, "INSERT INTO users (name, password) VALUES ('%s','%s')",
        username, password);

        if (mysql_query(conn, query) != 0)
        {
            fprintf(stderr, "%s\n", mysql_error(conn));
            exit(3);
            mysql_close(conn);
        }
        else
        {
            strcpy(answer, "Te-ai inregistrat cu succes\n");
            mysql_close(conn);
        }
    }
    else
    {
        strcpy(answer, "Exista deja un utilizator cu acest nume!\n");
        mysql_close(conn);
    }
    write(desc, answer, strlen(answer));
}
```

Comanda pentru trimis mesaje

```
void sendMessage(int desc, thData th, char buffer[])
{
    printf("am intrat in sendmsg\n");
    char nume[100] = " ";
    char comanda[100] = " ";
    char messageRead[500] = " ";
    char warning[5000] = " ";
    char toSend[1000] = " ";
    int clientDesc;

    sscanf(buffer, "%s %s", comanda, nume);
    printf("%s\n", nume);

    if (verifyUser(nume) == 1)
    {
        printf("am verificat\n");
        if (isOnline(nume) == true)
        {
            write(desc, "esteOnline", 10);

            read(desc, messageRead, sizeof(messageRead));

            sprintf(toSend, "Mesaj de la %s: ", getNameById(th.idUser));

            strcat(toSend, messageRead);

            printf("%s\n", messageRead);

            for (int i = 0; i < clientNumbers; i++)
            {
                if (clients[i]->idUser == getUserId(nume))
                {
                    clientDesc = clients[i]->thDesc;
                    // printf("%i\n", clientDesc);
                    // printf("Am gasit\n");
                    write(clientDesc, toSend, strlen(toSend));
                    insertOnlineMesssageIntoDataBase(th.idUser, getUserId(nume), messageRead);
                    printf("Am trimis\n");
                }
            }
        }
        else if (isOnline(nume) == false)
        {
            sprintf(warning, "Utilizatorul '%s' nu este online.
                   \nMesajul trimis o sa il primeasca cand se conecteaza!", nume);
            write(desc, warning, strlen(warning));

            read(desc, messageRead, sizeof(messageRead));

            printf("%s\n", messageRead);

            insertOfflineMesssageIntoDataBase(th.idUser, getUserId(nume), messageRead);
        }
    }
    else
    {
        sprintf(warning, "Nu exista niciun utilizator cu numele '%s'.", nume);
        write(desc, warning, strlen(warning));
    }
}
```

Fig.1 Diagrama Use Case

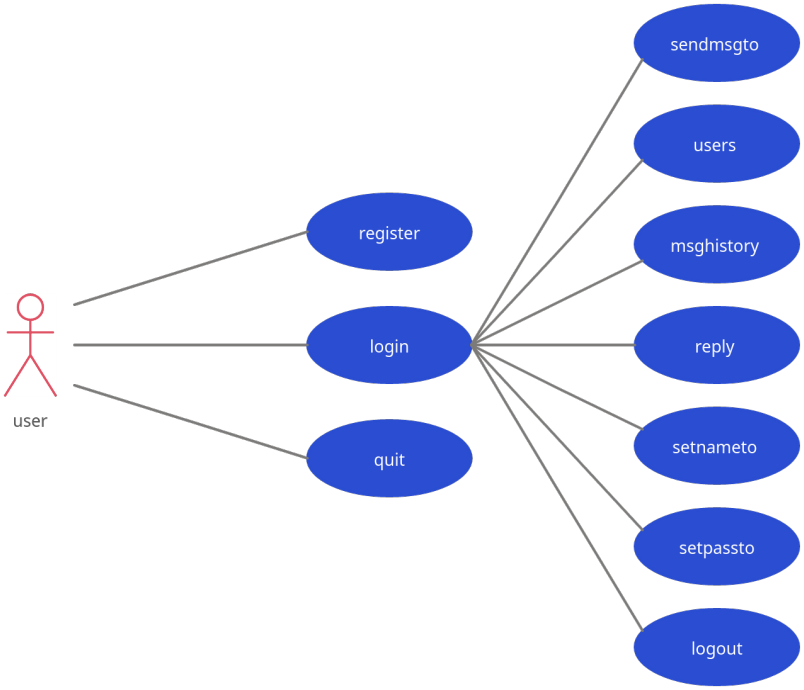


Fig.2 Structura baza de date

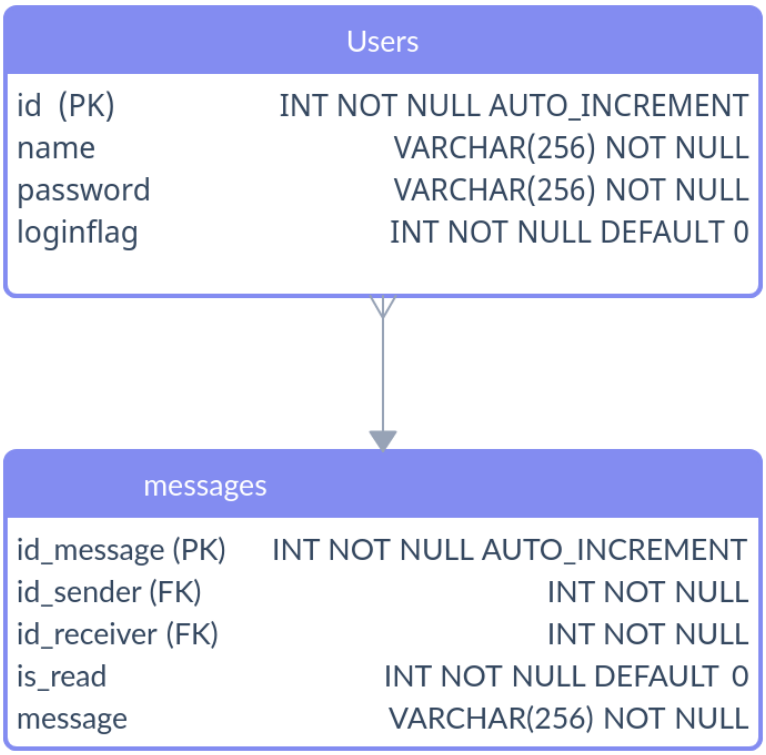
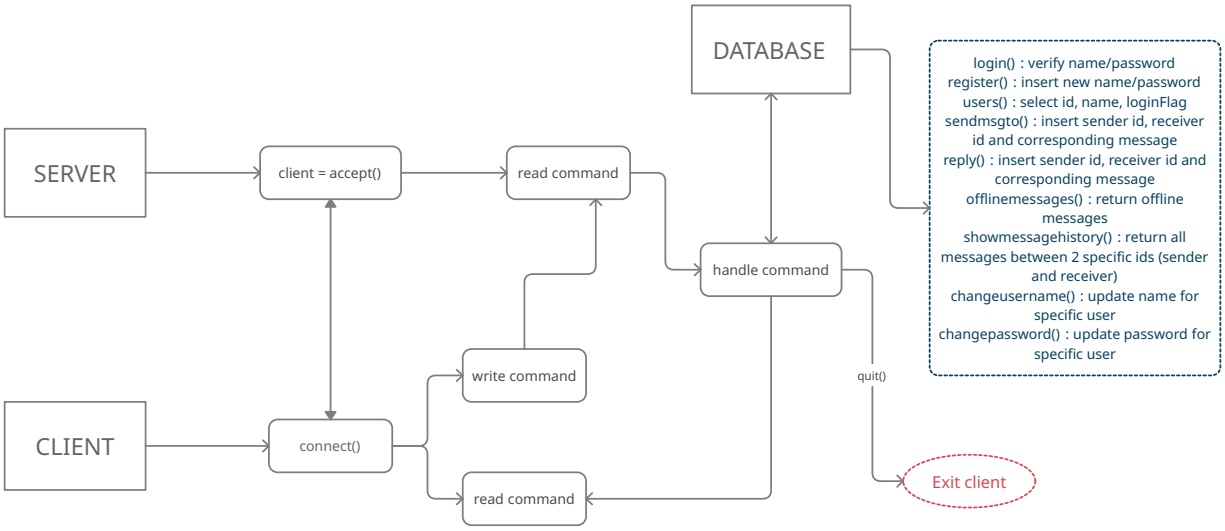


Fig.3 Structura programului



5 Concluzii

- Un element care ar face aplicatia mult mai atractiva si interactiva ar fi implementarea unei interfate grafice.
- Criptarea mesajelor si a parolelor este un lucru esential, securitatea fiind un lucru foarte important la o aplicatie de genul acesta.
- Implementarea unei liste de user favoriti pentru a comunica mult mai usor cu acestia.
- Implementarea unei sistem de a bloca utilizatorii cu care nu vrei sa interactionezi.
- Implementarea unei functionalitati de a putea trimite si fisiere pe langa mesaje.

6 Bibliografie

[1] `https://profs.info.uaic.ro/~computernetworks/index.php`

[2] `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh.c`

[3] `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c`

[4] `https://dev.mysql.com/doc/c-api/8.0/en/`