REȚELE DE CALCULATOARE
Facultatea de Informatică Iași
TEMA 2
OFFLINE MESSENGER

Mititelu Andi (2B4)

January 9, 2023

1 Introducere

Proiectul Offline Messenger se bazează pe comunicarea de tip client/server prin intermediul căreia avem posibilitatea de a trimite mesaje între utilizatori. Pentru a putea trimite un mesaj, utilizatorul trebuie să fie logat, iar userii offline vor primi mesajele în momentul reconectării la server. Clienții au posibilitatea de a se loga cu un username și o parolă, care vor fi stocate într-o bază de date, după care vor avea acces la mai multe opțiuni precum: să scrie mesaje către un utilizator, să poată vedea istoricul conversațiilor cu alți useri, să poata da reply specific la un mesaj primit etc. Fiecare utilizator se poate deconecta de la server oricând prin funcția "logout".

2 Tehnologii utilizate

- Am ales să folosesc ca protocol de comunicare TCP, deoarece asigură reliability în timpul transmiterii:
  - Toate pachetele de date ajung la destinație, fără pierdere de informație, un lucru esențial într-o aplicație de mesagerie.
  - Toate pachetele de date sunt reasamblate în ordine - luând cazul unui mesaj, nu am vrea literele amestecate.
- Gestionarea proceselor este realizată prin fire de execuție(thread-uri). Ele pot fi văzute ca un program în execuție fără spațiu de adresă proprie și, astfel, costurile creării și managementului proceselor este mai mic decât în cazul implementării cu mecanismul fork().
- Pentru stocarea datelor am folosit baza de date SQLite, prin care putem salva username-ul si parola utilizatorilor, putem verifica autenticitatea username-ului și putem gestiona istoricul mesajelor.
- Limbaj utilizat: C/C++.

3 Arhitectura aplicației

- **Register:** Utilizatorul va apela comanda */register* pentru a se autentifica. El va trebui să introducă un username și o parolă, iar, dacă username-ul nu există deja, datele vor fi stocate în câmpul Users.
- **Login:** Utilizatorul va apela comanda */login* pentru a se loga, iar apoi va putea folosi funcțiile aplicației.
- **Send message:** User-ul va putea trimite mesaje către alti utilizatori existenți prin comanda */message* $<$ *username* $><$ *text* $>$.

- **Reply:** User-ul va putea răspunde la un mesaj specific cât timp se află în conversație cu un alt utilizator $/reply < message >$.
- **Unseen messages:** User-ul va putea vedea mesajele pe care le-a primit cât timp a fost offline prin comanda $/showOffline$.
- **Show history:** Utilizatorul va putea vedea istoricul conversatiilor cu un alt utilizator prin comanda $/history$.
- **Show online users:** Userul poate vedea lista cu utilizatorii conectati prin comanda $/showOnline$.
- **Show offline users:** Userul poate vedea lista cu utilizatorii offline prin comanda $/showOffline$.
- **Logout:** Prin comanda $/logout$ utilizatorul se va putea deconecta de la server.
- **Exit:** Prin comanda $/exit$ utilizatorul va închide aplicația.

Diagrama pentru structura programului:

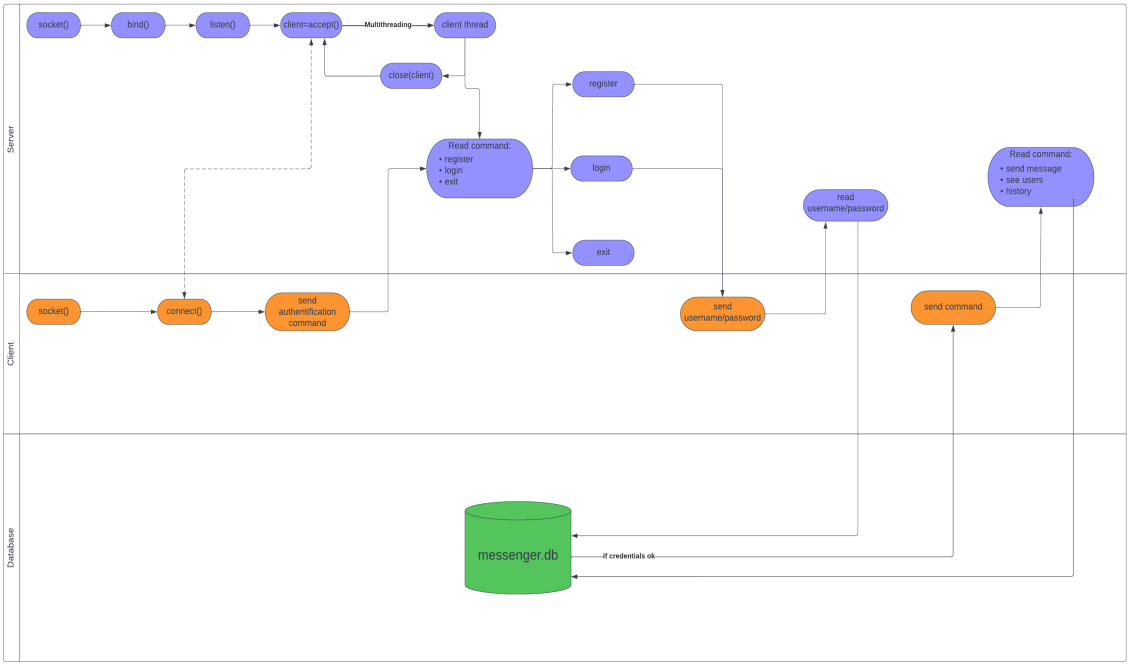
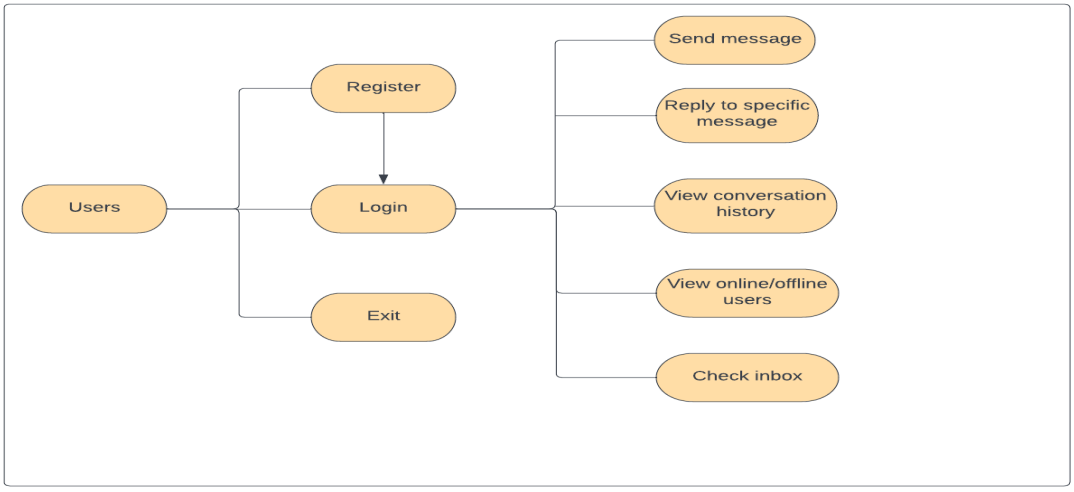


Diagrama Use Case:



4 Detalii implementare

Comunicarea dintre server și client se realizează prin socket-uri.

```c
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
{
  perror ("Eroare la socket...\n");
  exit(1);
}
```

Pentru concurența serverului am folosit thread-uri astfel: pentru fiecare client conectat un thread se va ocupa de comenzile lui.

```c
pthread_t th[100];    //Identificatorii thread-urilor care se vor crea
typedef struct thData{
  int idThread; //id-ul thread-ului tinut in evidenta de acest program
  int cl; //descriptorul intors de accept
}thData;

static void *treat(void *); /* functia executata de fiecare thread ce realizeaza comunicarea cu clientii */
void raspunde(void *);
```

Pentru gestionarea datelor din aplicație am folosit o bază de date în SQLite, care va fi de ajutor pentru stocarea datelor de logare ale utilizatorilor și păstrarea thread ID-ului si al login-ului. Un exemplu în care folosim bazele de date e la înregistrarea unor utilizatori noi:

```c
if(checkUser(username) == 0)
{
  printf("checkUser: %d\n", checkUser(username));
  //trimitem mesaj catre client ca user-ul e bun si poate introduce parola
  exista = 0;
  if(write(tdL.cl, &exista, sizeof(int)) <= 0)
  {
    perror("[server]Eroare la trimis user existent\n");
  }
  if(read(tdL.cl, password, sizeof(password)) <= 0)
  {
      perror("[server]Eroare la read la password\n");
  }
  fflush(stdout);
  printf("[server]Am ajuns sa primim parola: %s\n", password);

  exista = 0;
  sqlite3 *db;
  char *err_msg = 0;
  int rc = sqlite3_open("messenger.db", &db);
  if (rc != SQLITE_OK)
  {
    fprintf(stderr, "Cannot open database: %s\n", sqlite3_errmsg(db));
    sqlite3_close(db);
    return 1;
  }
```

```
  char* sql;
  sql = (char*) malloc(2048);
  strcpy(sql, "DROP TABLE IF EXISTS Users;"
              "CREATE TABLE Users(Name TEXT, Password TEXT);"
              "INSERT INTO Users VALUES ('dqdq', 'abcd12');"
  );
  int name_len = sizeof(username);
  int pass_len = sizeof(password);
  int insert_len = sizeof("INSERT INTO Users VALUES ('', '');");

  snprintf(sql, name_len + pass_len + insert_len, "INSERT INTO Users VALUES ('%s', '%s');", username, pas
  rc = sqlite3_exec(db, sql, 0, 0, &err_msg);

  if (rc != SQLITE_OK )
  {
    fprintf(stderr, "SQL error: %s\n", err_msg);
    sqlite3_free(err_msg);
    sqlite3_close(db);
    return 1;
  }
  free(sql);
  sqlite3_close(db);
  break;
}
else
{
  printf("[server]Utilizatorul exista deja\n");
  //aici trimitem la client sa introduca un nou username
```

Cu ajutorul bazei de date pot afla daca un utilizator este logat sau nu in functie de valoare campului ”login”. De asemenea, cu ajutorul campului ”td” clientul poate trimite mesaj catre un alt user astfel: obtine thread ID-ul corespunzator cu username-ul catre care vrea sa trimita mesajul apoi apeleaza primita *write*().

```
int getTh(char username[256])
{
  sqlite3 *db;
  char *err_msg = 0;
  sqlite3_stmt *res;
  username[strlen(username)] = '\0';

  //sqlite3_config(SQLITE_CONFIG_SINGLETHREAD);
  int rc = sqlite3_open("messenger.db", &db);
  rc = sqlite3_exec(db, "BEGIN TRANSACTION", NULL, NULL, &err_msg);
  if (rc != SQLITE_OK)
  {

      fprintf(stderr, "Cannot open database: %s\n",
              sqlite3_errmsg(db));
      sqlite3_close(db);

      return -2;
  }
```

```
strcpy(sql, "SELECT td from Users WHERE Name = ?");
rc = sqlite3_prepare_v2(db, sql, -1, &res, 0);

if (rc == SQLITE_OK)
{
  sqlite3_bind_text(res, 1, username, -1, SQLITE_STATIC);
}
else
{
  fprintf(stderr, "Failed to execute statement: %s\n", sqlite3_errmsg(db));
}
int step = sqlite3_step(res);

if (step == SQLITE_ROW)
{
  printf("[getTh] ALL GUD!\n");
  //return 1;
}
else
{
  fprintf(stderr, "[getTh]Failed to execute in Th...\n");
  return 0;
}
int value = sqlite3_column_int(res, 0);
```

Clientul poate verifica si ce utilizatori sunt online si logati astfel:

```
if(strcmp(cmdReceived, "|listUsers") == 0)
{
    sqlite3* db;
    sqlite3_stmt* stmt;
    int rc;
    char *err_msg = 0;
    // Open the database
    rc = sqlite3_open("messenger.db", &db);
    rc = sqlite3_exec(db, "BEGIN TRANSACTION", NULL, NULL, &err_msg);
    if (rc != SQLITE_OK) {
      fprintf(stderr, "Error opening database: %s\n", sqlite3_errmsg(db));
      return 1;
    }
    // Prepare the SQL statement
    rc = sqlite3_prepare_v2(db, "SELECT * FROM users where login='1';", -1, &stmt, NULL);
    if (rc != SQLITE_OK) {
      fprintf(stderr, "Error preparing statement: %s\n", sqlite3_errmsg(db));
      return 1;
    }
    // Iterate over the result set and extract the text value of the first column
    char* names = malloc(1);
    names[0] = '\0';
```

```
while (sqlite3_step(stmt) == SQLITE_ROW) {
  char* name = (char*)sqlite3_column_text(stmt, 0);
  names = realloc(names, strlen(names) + strlen(name) + 2);
  strcat(names, name);
  strcat(names, " ");
}

sqlite3_finalize(stmt);
sqlite3_exec(db, "END TRANSACTION", NULL, NULL, NULL);
sqlite3_close(db);

if(names != NULL)
{
  names[strlen(names) - 1] = '\0';
  for(int i = 0; i < strlen(names); i++)
  {
    if(names[i] == ' ')
      names[i] = '|';
  }
  char output[500] = "";
  strcpy(output, names);
  if(write(tdL.cl, output, sizeof(output)) <= 0)
  {
    perror("[server|onlineUsers]Eroare la write catre client...\n");
  }
}
else
{
  printf("[server]Niciun user online!\n");
  char none[100] = "";
  strcpy(none, "Niciun user online!");
  if(write(tdL.cl, none, sizeof(none)) <= 0)
  {
    perror("[server|onlineUsers(none)]Eroare la write catre client...\n");
  }
}
free(names);
```

5 Concluzii

- O imbunatatire importanta este implementarea posibilitatii de a trimite mesaje catre utilizatori offline, specificata de altfel in enunt, care ulterior la logare sa fie notificati.
- O îmbunătățire a aplicației ar fi adăugarea unei interfețe grafice.
- Un alt feature ar putea fi adăugarea opțiunii de creare de grupuri, cu username-uri personalizate grupurilor respective.
- Un update bazei de date, adica criptarea ei, ar duce la securizarea conturilor utilizatorilor.

6 Bibliografie

- https://zetcode.com/db/sqlitec/
- https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
- https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
- https://profs.info.uaic.ro/~eonica/rc/lab11.html
- https://profs.info.uaic.ro/~ioana.bogdan/
- https://profs.info.uaic.ro/~andreis/wp-content/uploads/2022/12/Laboratorul-12.pdf