Monitorizarea traficului(A)

Damian Alexandru, grupa B5, anul 2

Facultatea de informatica Iasi

Keywords: TCP · client-server · threads

1 Introducere

Motivatie Am ales sa fac acest proiect deoarece am vrut sa aprofundez conceptele de retelistica. Consider ca acest proiect ma va ajuta sa inteleg mai bine cum sunt implementate sistemele de gestiune a traficului, sau aplicatiile folosite pentru navigare.

Voi realiza aplicatia in limbajul C cu o interfata grafica pentru client realizata in Qt creator sau Glade.

2 Tehnologiile utilizate

In acest proiect voi folosi protocolul TCP/IP deoarece vreau sa fiu sigur ca toti clientii vor primi notificarile pentru evenimente, accidente din zonele in care se afla. Prin utilizarea protocolului TCP/IP, clientul va crea o conexiune stabila cu serverul, astfel transferul de informatii va fi garantat.

Pentru a permite accesul mai multor clienti la server in acelasi timp, voi crea un server TCP concurent. Pentru fiecare client conectat, voi crea un nou thread in server.

De asemenea, la proiect ma voi folosi si de o baza de date SQlite in care voi retine datele de logare a clientilor, dar si preferintele lor (informatii despre vreme, evenimente sportive, preturi pentru combustibili la statiile peco). Pentru a utiliza acesta baza de date, voi folosi libraria sqlite3.h .

Pentru a realiza o interfata grafica pentru client, voi folosi libraria gtk si glade sau voi folosi qt creator.

3 Arhitectura aplicatiei

Serverul va astepta conexiuni din partea clientilor. Pentru fiecare client, serverul va obtine 2 file-descriptori prin care va comunica cu acel client, si va crea un

thread doar pentru acel client. Deci pentru fiecare client, in server va exista cate un thread.

Primul fd, va fi pentru comunicarea datelor de logare/inregistrare, si comunicarea optiunilor si incidentelor trimise de client catre server. Al doilea fd, va fi folosit pentru a realiza broadcastul. Cand un client va trimite o informatie despre un accident, serverul va inregistra aceasta informatie, si va seta o variabila accesibila fiecarui thread astfel incat in fiecare thread va sti ca va trebui sa trimita informatia clientului sau.

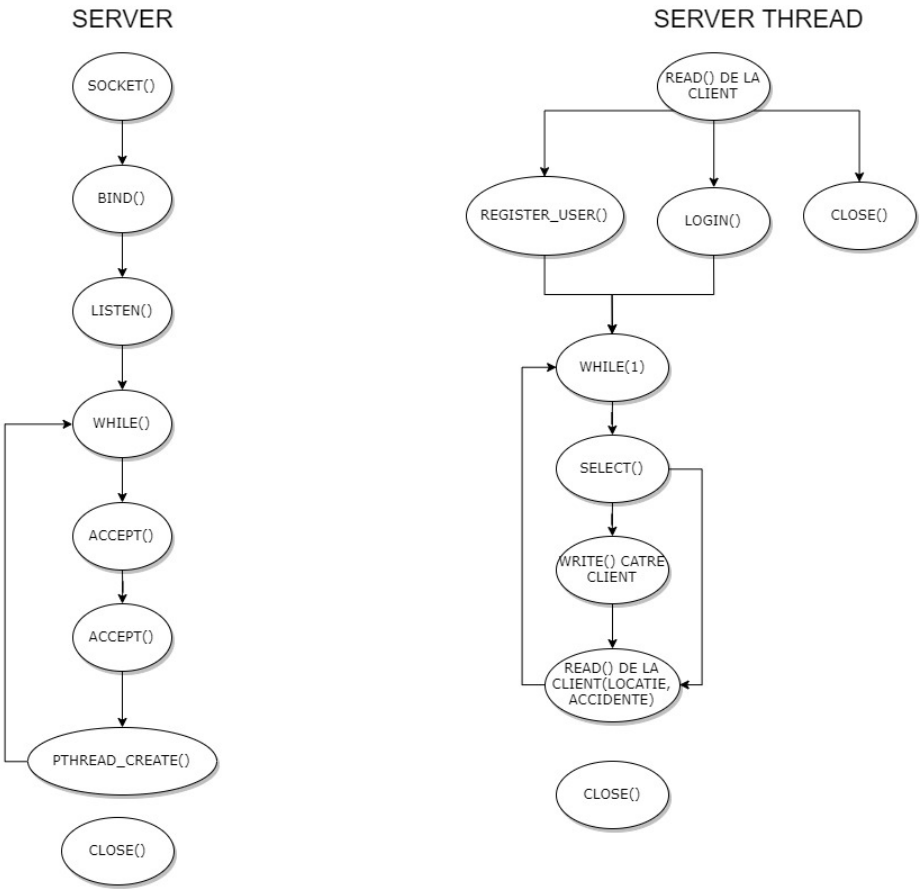

Fig. 1: Diagrama server

Clientul va avea un thread si 2 socketi. Pe primul socket va comunica cu serverul datele de logare, si va introduce introduce date in caz de accident. Al

doilea socket, va trimite date la server despre locatia si viteza clientului, si va primi de la server informatii despre limita de viteza, incidente (maracte de alti utilizatori) si evenimentele din acea zona.

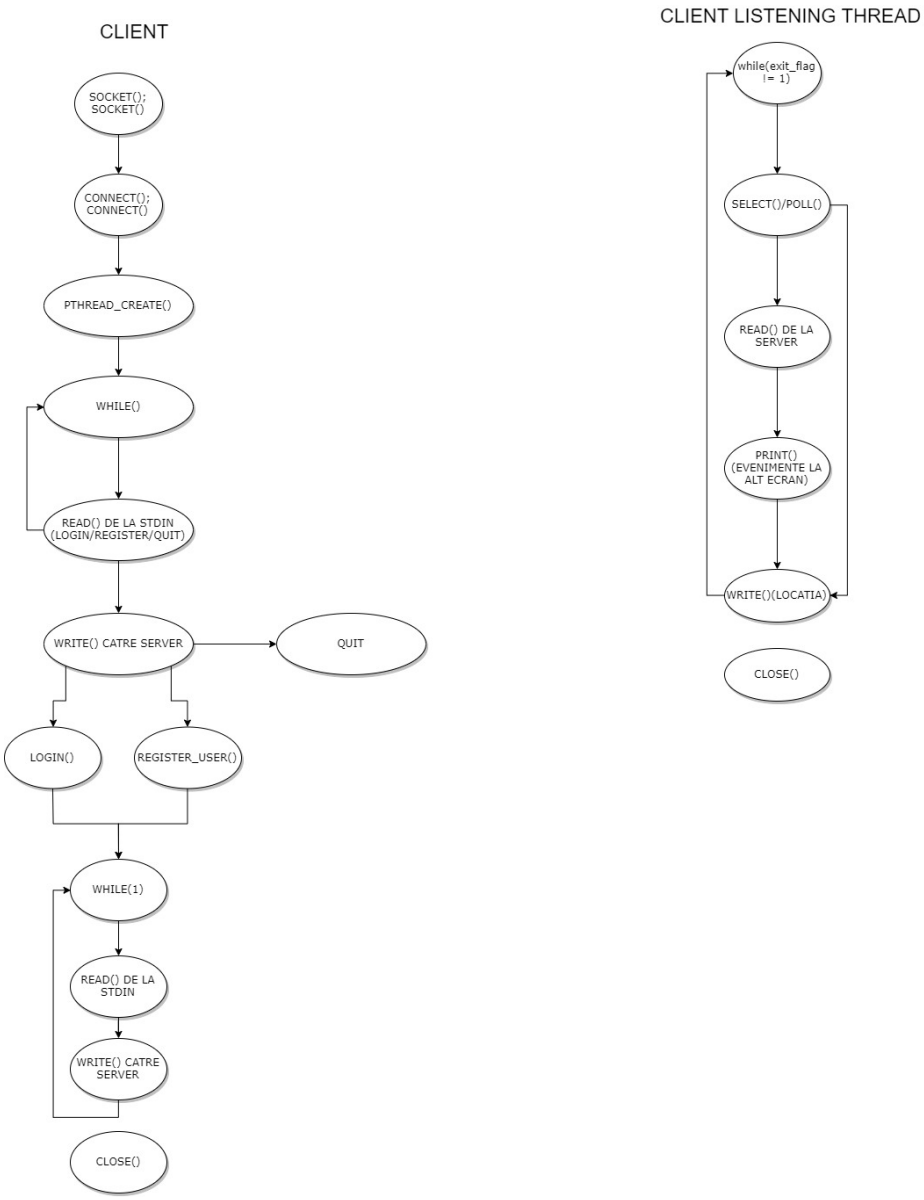

Fig. 2: Diagrama client

Clientul va avea 2 ecrane diferite, unul in care introduce inputul de la tastatura, si altul care va afisa evenimentele/incidente/limitari de viteza primite de la server in functie de zona in care se afla.

Pentru simularea unei harti, voi folosi o matrice, in care fiecare pozitie (linie, coloana) va fi (y, x) intr-un sistem cartezian. Fiecare strada din oras va avea o cate un vector de coordonate (x,y). Reteaua stradala va fi modelata sub forma unui graf, in care muchiile vor fi strazile, iar nodurile vor fi intersectiile. Clientii se vor deplasa de la un punct (x, y) la alt punct (p, q). Pentru a determina un drum de lungime minima, ma voi folosi de un algoritm pentru determinarea unui drum de lungime minima. Zonele cu restrictie de viteza, blocaje si accidente vor fi retinute in server sub forma unor vectori de coordonate.

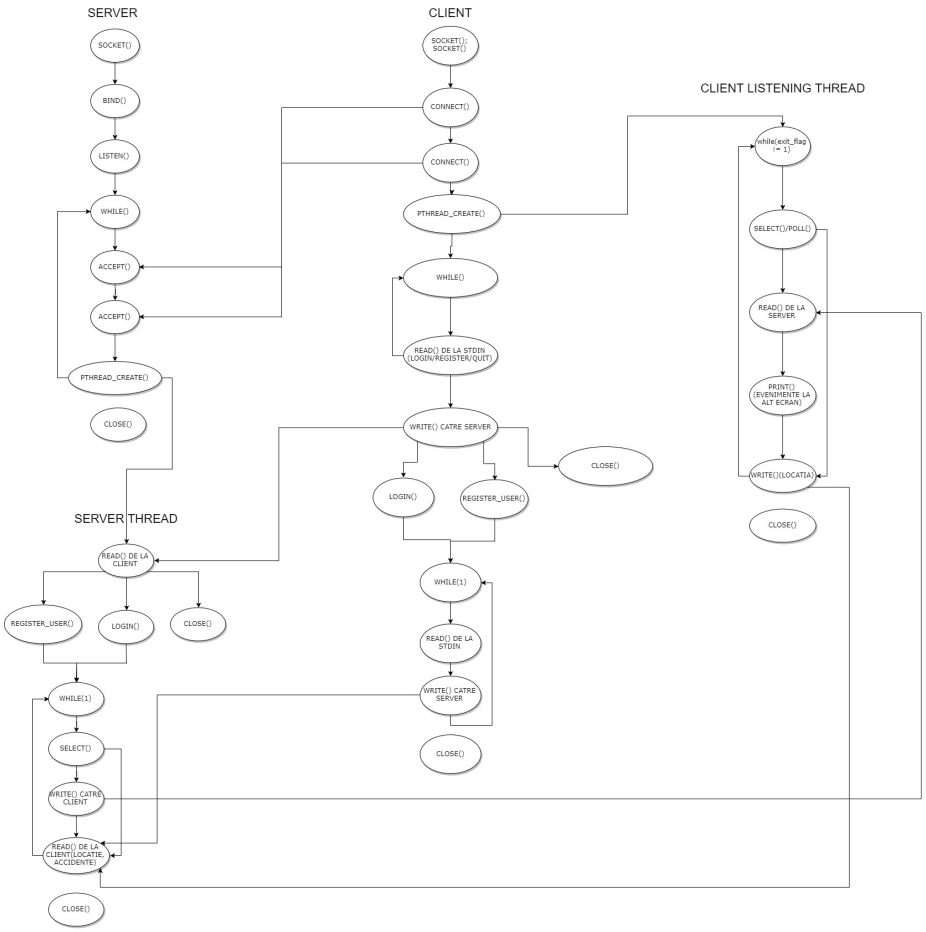

Fig. 3: Diagrama generala a aplicatiei

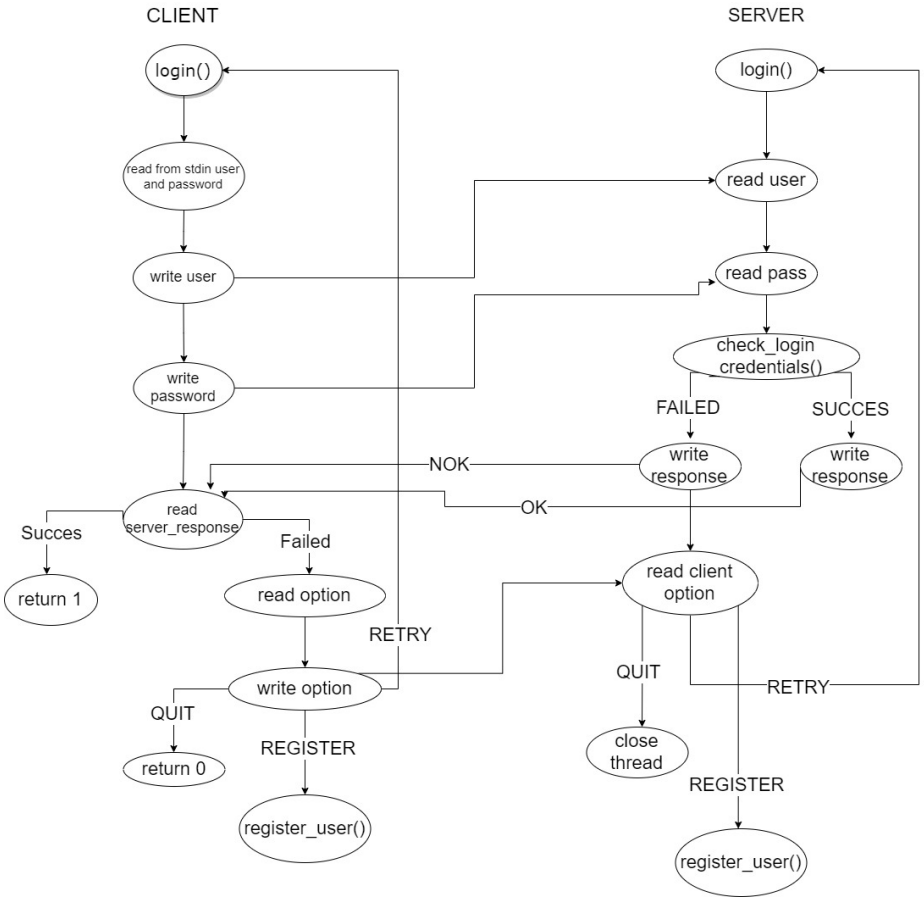

Fig. 4: Functia login

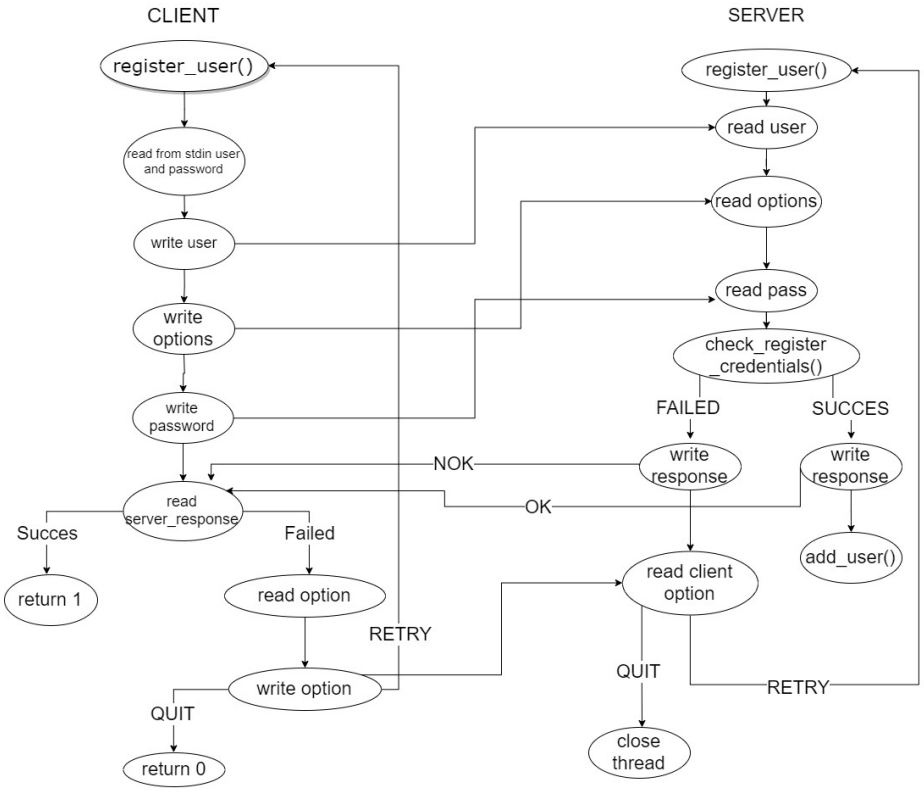

Fig. 5: Functia register

4 Detalii de implementare

Clientul va avea de ales intre register, login si quit. Daca se va inregistra, acesta va trebui sa isi aleaga si evenimentele pe care vrea sa le primeasca(informatii despre vreme, evenimente sportive, preturi pentru combustibili la statiile peco). Dupa ce conectarea s-a realizat cu succes, clientul va putea sa introduca accident sau inchide. Daca introduce accident, va fi intrebat de locatia accidentului si informatia va fi retinuta de server si distribuita catre alti clienti din acea zona.

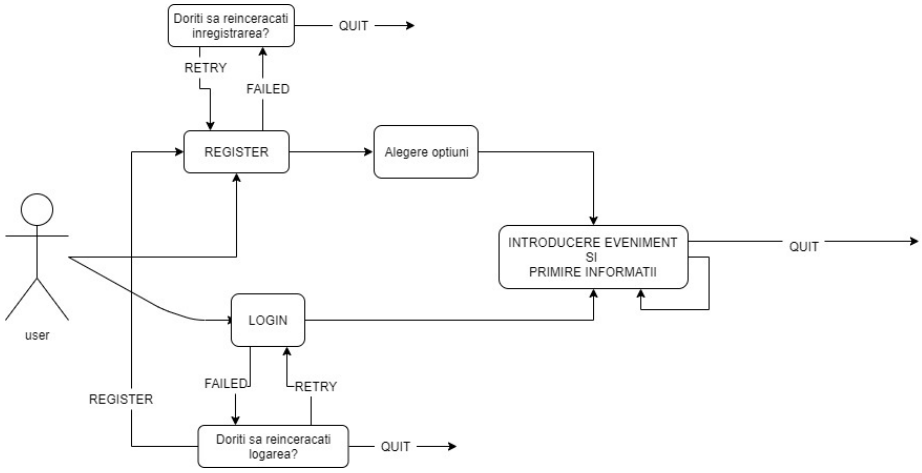

Fig. 6: Cum va folosi clientul aplicatia

Functia de login din client

```
1 int login(int sd){//1 == succes, 0 == quit
2     char* user = (char*)malloc(50);
3     char* pass = (char*)malloc(50);
4     printf("user : ");
5     fgets(user, 50, stdin);
6     user[strlen(user) - 1] = '\0';
7     printf("\n password : ");
8     fgets(pass, 50, stdin);
9     pass[strlen(pass) - 1] = '\0';
10    if(write(sd, user, 50) < 0){
11        perror("Eroare la write");
12        return 0;
13    }
14    if(write(sd, pass, 50) < 0){
15        perror("Eroare la write");
16        return 0;
17    }
18    char server_response[100];
19    read(sd, server_response, 100);
20    if(strcmp(server_response, "OK") == 0){
21        printf("\nLOGIN SUCCESFULL\n");
22        return 1;
23    }else{
24        printf("\n Login failed. \n");
```

```
25          printf("Retry to login or register? Type your
                option[retry/register/quit] ");
26          char * action = (char*)malloc(50);
27          fgets(action, 50, stdin);
28          action[strlen(action) - 1] = '\0';
29          while(strcmp(action, "retry") != 0 &&
                strcmp(action, "quit") != 0 && strcmp(action,
                "register") != 0 ){
30              printf("Please type retry or quit or
                    register.\n");
31              fgets(action, 50, stdin);
32          }
33          if(strcmp(action, "quit") == 0){
34              if(write(sd, action, 10) < 0){
35                  perror("Eroare la write");
36                  return 0;
37              }
38              return 0;
39          }else if(strcmp(action, "retry") == 0){
40              if(write(sd, action, 10) < 0){
41                  perror("Eroare la write");
42                  return 0;
43              }
44              return login(sd);
45          }else{
46              if(write(sd, action, 10) < 0){
47                  perror("Eroare la write");
48                  return 0;
49              }
50              return register_user(sd);
51          }
52      }
53      return 0;
54  }
```

Functia de register user din client

```
1  int register_user(int sd){//1 == succes, 0 == quit
2      char* user = (char*)malloc(50);
3      char* pass = (char*)malloc(50);
4      printf(" user : ");
5      fgets(user, 50, stdin);
6      user[strlen(user) - 1] = '\0';
7      printf("\n password : ");
8      fgets(pass, 50, stdin);
```

```
 9      pass[strlen(pass) - 1] = '\0';
10      if(write(sd, user, 50) < 0){
11          perror("Eroare la write");
12          return 0;
13      }
14      if(write(sd, pass, 50) < 0){
15          perror("Eroare la write");
16          return 0;
17      }
18      char server_response[100];
19      read(sd, server_response, 100);
20      if(strcmp(server_response, "OK") == 0){
21          printf("\nREGISTER SUCCESFULL\n");
22          return 1;
23      }else{ // not ok
24          printf("\n REGISTER failed. \n");
25          printf("Retry to register? Type your
                option[retry/quit] ");
26          char* action = (char*)malloc(50);
27          fgets(action, 50, stdin);
28          action[strlen(action) - 1] = '\0';
29          while(strcmp(action, "retry") != 0 &&
                strcmp(action, "quit") != 0){
30              printf("Please type retry or quit \n");
31              fgets(action, 50, stdin);
32          }
33          if(strcmp(action, "quit") == 0){
34              if(write(sd, action, 10) < 0){
35                  perror("Eroare la write");
36                  return 0;
37              }
38              return 0;
39          }else if(strcmp(action, "retry") == 0){
40              if(write(sd, action, 10) < 0){
41                  perror("Eroare la write");
42                  return 0;
43              }
44              return register_user(sd);
45          }
46      }
47      return 0;
48  }
```

Functia register user din server

```
1 void register_user(int sd, int id){
```

```
2       /// check ///
3       while(1){
4           char user[INPUT_SIZE];
5           char pass[INPUT_SIZE];
6           read(sd, user, INPUT_SIZE);
7           read(sd, pass, INPUT_SIZE);
8           int check_code =
                check_register_credentials(user, pass);
9           printf("code = \%d\n", check_code);
10          if(check_code == 1){
11              write(sd, "NOT OK", 7);
12              //vezi ce a trimis clientul
13              char option[100];
14              read(sd, option, 100);
15              if(strcmp(option, "quit") == 0){
16                  check_code = -1;
17                  break;
18              }else if(strcmp(option, "retry") == 0){
19                  continue; //bucla while va merge inca o
                        data
20              }
21          }else if(check_code == 0){//add user
22              add_user(user, pass);
23              printf("[thread \%d]Clientul \%s s-a
                    inregistrat cu succes!\n", id, user);
24              write(sd, "OK", 3);
25              break;
26          }
27      }
28  }
```

Functia check login credentials din server

```
1  int check_login_credentials(char*user, char*pass){// -1
       daca e eroare, 1 daca am gasit user + pass, 0 altfel
2       int fd = open("useri.txt", O_RDWR);
3       if(fd < 0){
4           perror("Eroare la deschidere fisier");
5           return -1;
6       }
7       int found = 0, k = 0;
8       char word1[110];
9       while(!found){//parcurg fisierul
10          unsigned char ch;
11          int codr = read(fd, &ch, 1);
```

```
12          if(codr == 0){  //end of file
13              break;
14          }else if(codr == -1){
15              perror("Eroare la citire");
16              close(fd);
17              return -1;
18          }
19          if(ch != ' ' && ch != '\n'){//obtinem user-ul de
                pe o linie
20              word1[k++] = ch;
21          }
22          if(ch == ' '){
23              word1[k] = '\0';
24
25              if(strcmp(word1, user) != 0){//daca e un
                    user diferit, sarim linia
26                  while(ch != '\n'){//parcurgem linia pana
                        la sfarsit
27                      read(fd, &ch, 1);
28                  }
29                  k = 0;
30              }else{ // daca am gasit user-ul verificam
                    parola
31                  int t = 0;
32                  char password[50];
33
34                  while(ch != '\n'){//obtinem parola
35                      read(fd, &ch, 1);
36                      password[t++] = ch;
37                  }
38                  password[t-1] = '\0';
39
40                  if(strcmp(pass, password) ==
                        0){//verificam parola
41                      close(fd);
42                      return 1;
43                  }
44                  k = 0;
45              }
46          }
47      }
48      close(fd);
49      return found;
50  }
```

Un thread din server

```
1 void* routine(void*arg){
2     thData thread = *((thData*)arg);
3     printf("[thread \%d] Vom comunica cu clientul cu fd
          \%d\n", thread.thread_id, thread.client_fd);
4     int client_fd = thread.client_fd;
5     char client_option[100];
6     read(client_fd, client_option, 100);
7     printf("[thread \%d]Am primit >\%s< de la client\n",
          thread.thread_id, client_option);
8     if(strcmp(client_option, "register") == 0){
9         register_user(client_fd, thread.thread_id);
10    }else if(strcmp(client_option, "login") == 0){
11        login(client_fd, thread.thread_id);
12    }else if(strcmp(client_option, "quit") == 0){
13        printf("[thread \%d] Inchidere thread.\n",
              thread.thread_id);
14        close((uintptr_t)arg);
15        return NULL;
16    }
17    //TODO: broadcast cu thread.broadcast_fd
18    printf("[thread \%d]Inchidem thread.\n",
          thread.thread_id);
19    close((uintptr_t)arg);
20    return NULL;
21 }
```

5 Concluzii

Acest proiect poate fi imbunatatit prin implementarea unei interfete grafice care sa afiseze o harta actualizata in timp real. De asemenea, daca s-ar putea modifica optiunile alese de clienti si dupa ce s-au inregistrat, utilizatorii ar avea parte de o experienta mai buna a aplicatiei.

Pentru a imbunatati viteza serverului, as putea sa apelez functia accept() in threaduri, adica sa fac un threadpoll, sau as putea sa implementez un server TCP prethreaded.

6 Bibliografie

References

1. https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm
2. https://www.sqlite.org/cintro.html
3. https://www.sqlite.org/cintro.html
4. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
5. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
6. https://profs.info.uaic.ro/ computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
7. https://www.qt.io/
8. https://glade.gnome.org/
9. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php