# Monitorizarea traficului (A)

Haiura Andreea-Isabela

Universitatea Alexandru Ioan Cuza

## 1 Introducere

Am ales să redactez raportul tehnic pentru proiectul **Monitorizarea traficului (A)** deoarece cred că este un exercițiu complex din care aș avea foarte multe de învățat.

La prima citire a cerinței mi s-a părut asemănator cu o aplicație pe care o utilizez destul de des ce ajută șoferii în trafic: Waze, însă mult simplificată. Proiectul presupune proiectarea unui server concurent, a unui client și a unei baze de date. Clienții se conectează la server și pot alege diferite opțiuni pe care serverul să le trimită: vreme, sport și prețul combustibilului. Clientul o să fie atenționat în cazul în care depășește viteza, dar poate și să raporteze diferite accidente ce vor fi transmise către toți utilizatorii conectați la server.

Doresc să implementez acest proiect din pură curiozitate, pentru a înțelege anumite funcționalități ale aplicației respective.

## 2 Tehnologii utilizate

Modelul TCP conferă siguranță deoarece nu există riscul de a pierde date la transmiterea mesajelor, pe când în modelul UDP acest lucru este posibil. Totuși, UDP este mult mai rapid decât TCP, mai ales atunci când se transmit foarte multe pachete de date.

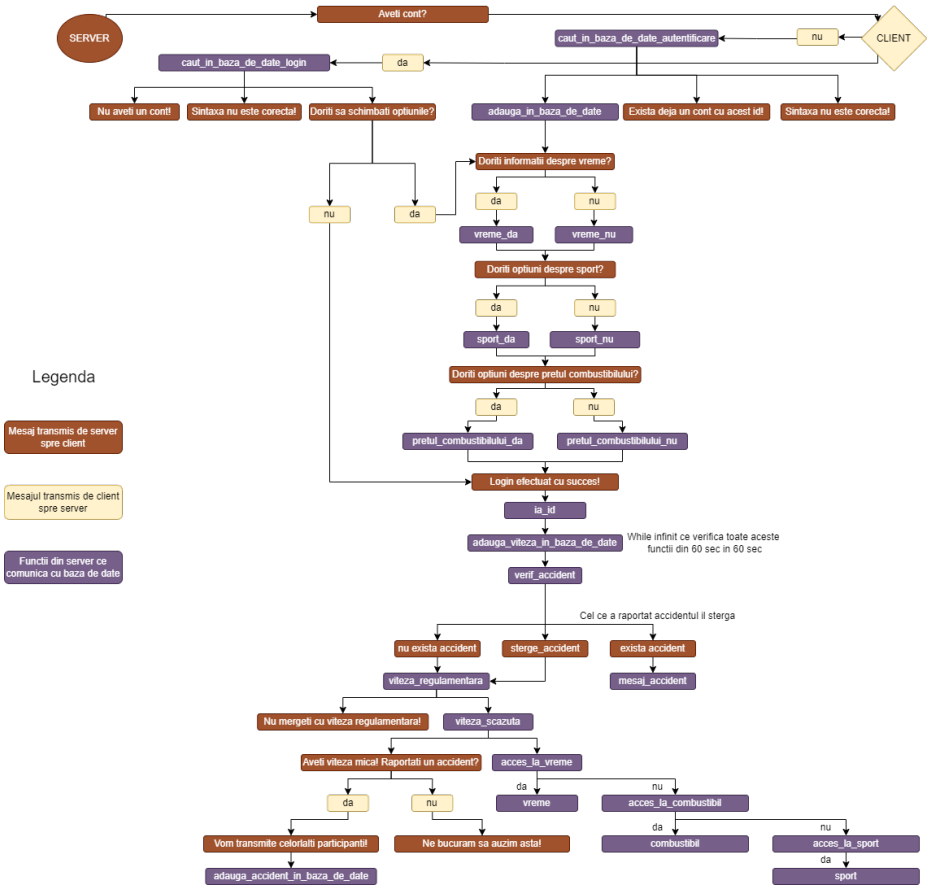
În cadrul proiectului am ales să utilizez modelul TCP deoarece la nivelul de înțelegere a aplicației și la implementarea minimală este mai important ca pachetul de date să fie sigur astfel încât să mă pot concentra mai mult pe aplicație. Având în vedere că datele transferate de la client la server și invers, nu sunt foarte multe, modelul ales nu reprezintă o piedică a rapidității în transmiterea mesajelor. Modelul UDP este mai rapid, dar există posibilitatea pierderii de date, iar acest lucru nu ar trebui să se întâmple în cadrul proiectului deoarece în cazul unui accident raportat, dacă serverul nu anunță ceilalți clienți de respectivul eveniment, strada se va bloca, iar traficul va fi îngreunat.

## 3 Arhitectura aplicației

Proiectul ales are un server la care se pot conecta unul sau mai mulți clienți în mod concurent. Clientul trebuie să folosească un cont creat ulterior sau să creeze unul pentru a se autentifica și pentru a putea pătrunde în opțiunile aplicației.

Utilizatorul poate alege să primească informații despre vreme, sport și prețul combustibilului, iar viteza va fi trimisă automat din 60 de secunde în 60 de secunde. În cazul în care utilizatorul depășește viteza legală pe acea porțiune de drum, va fi înștiințat. Dacă apare un accident, acesta poate fi sesizat de către utilizatori, urmând ca toți ceilalți să primească mesajul de înștiințare corespunzător.

În proiect am folosit o baza de date creată cu MySql în care am adăugat un tabel cu utilizatorii aplicației, deci în server am introdus biblioteca necesară pentru a accesa baza de date din programul $C$, și anume: $\#include < mysql.h >$. Am creat tabele în care am reținut străzile cu restricțiile respective de viteză, informații referitoare la vreme, sport și prețul combustibilului.



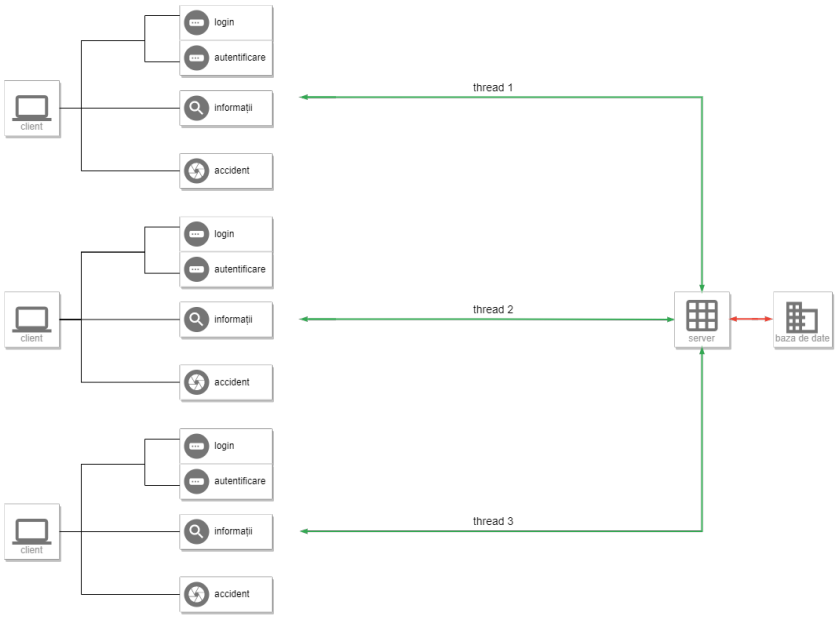

În diagrama de mai sus se observă cum serverul comunică cu clientul și cu baza de date, iar clientul și baza de date comunica doar cu serverul.

Serverul o să întrebe inițial dacă clientul are un cont, iar în caz contrat are posibilitatea să creeze unul. Conturile sunt căutate în baza de date, unde cheia primară este reprezentată de un id. Dacă clientul nu are un cont și creează unul, va fi intrebat daca dorește informații referitoare la sport, vreme și prețul combustibilului unde acesta răspunde cu da sau nu. Dacă clientul se loghează în cont este întrebat dacă dorește să schimbe aceste opțiuni. Dacă răspunsul este da, este întrebat încă o dată cu ce opțiuni dorește să rămână, iar în caz contrar, dacă scrie nu, opțiunile nu se modifică.

În cazul în care clinetul s-a logat cu succes, acestuia i se va lua automat viteza și va fi înștiințat în cazul în care depășește limita legală pe acea porțiune de drum. Informațiile despre vreme, sport și prețul combustibilului se vor afla în baza de date, fiind preluate de server și transmise clientului. În cazul unui accident, clientul informează serverul, urmând ca serverul să informeze toți clienții conectați.

## 4 Detalii de implementare

Proiectul este implementat pe modelul client-server concurent folosind threaduri. Concurența este dată de threaduri deoarece fiecare client se conectează pe threaduri diferite ale procesului. Datorită acestui fapt se pot conecta mai mulți clienți odată, fiind serviți în mod concurent, după cum putem observa și în diagrama de mai jos.



- *int ia_id(char x[100])* : Returnează id-ul clientului din sintaxa cu care acesta se loghează, și anume: nume.prenume.id.

- *int caut_in_baza_de_date_login(char x[100])* : Caută în baza de date numele, prenumele și id-ul clientului și verifică și corectitudinea sintaxei. Dacă există un cont cu specificațiile date de către client returnează 1, în caz contrar returnează 0 și -1 în cazul unei erori la căutarea în baza de date.

- *int caut_in_baza_de_date_autentificare(char x[100])* : Caută în baza de date id-ul clientului și verifică și corectitudinea sintaxei. Dacă există deja un cont cu acel id funcția va returna 1, în caz contrar va returna 0 și -1 în cazul unei erori la căutarea în baza de date.

- *int adauga_in_baza_de_date(char x[100])* : Adaugă în baza de date contul cu numele, prenumele și id-ul furnizat de client.

- *int acces_la_vreme(int x)* : Verifică în baza de date dacă clientul dorește informații despre vreme.

- *int vreme_da(int x)* : Clientul dorește informații despre vreme și modificăm opțiunea în baza de date.

- *int vreme_nu(int x)* : Clientul nu dorește informații despre vreme și modificăm opțiunea în baza de date.

- *char* vreme()* : Returnează vremea din baza de date.

- *int acces_la_sport(int x)* : Verifică în baza de date dacă clientul dorește informații despre sport.

- *int sport_da(int x)* : Clientul dorește informații despre sport și modificăm opțiunea în baza de date.

- *int sport_nu(int x)* : Clientul nu dorește informații despre sport și modificăm opțiunea în baza de date.

- *char* sport()* : Returnează o știre despre sport din baza de date.

- *int acces_la_combustibil(int x)* : Verifică în baza de date dacă clientul dorește informații despre prețul combustibilului.

- *int pretul_combustibilului_da(int x)* : Clientul dorește informații despre prețul combustibilului și modificăm opțiunea în baza de date.

- *int pretul_combustibilului_nu(int x)* : Clientul nu dorește informații despre prețul combustibilului și modificăm opțiunea în baza de date.

- *char* combustibil()* : Returnează prețul combustibilului din baza de date.

- *int adauga_viteza_in_baza_de_date(int id, int viteza)* : Adaugă viteza în baza de date.

- *int viteza_regulamentara(int x)* : Adaugă strada în client și verifică dacă are viteza regulamentară pe porțiunea de drum pe care merge accesând tabelul *viteza* din baza de date.

- *int viteza_scazuta(int x)* : Dacă viteza cu care merge clientul este sub 20 $km/h$, serverul îl întreabă dacă dorește să raporteze un accident și totodată poate ieși din aplicație tastând comanda: *exit*.

- *int verif_accident(int x)* : Verifică dacă există accidente raportate în baza de date.

- *int adauga_accident_in_baza_de_date(int x)* : Adaugă id-ul clientului care a raportat accidentul în baza de date și strada pe care se află respectivul pentru a anunța ceilalți clienți cu privire la locul accidentului.

- *int sterge_accident(int x)* : Clientul care a adăugat accidentul în baza de date îl va șterge dupa 60 de secunde.

- *char* mesaj_accident()* : Returnează Mesajul pe care îl va transmite celorlalți participanți la trafic respre accidentul produs. Acesta va conține mesajul de atenționare și strada pe care s-a efectuat accidentul respectiv.

- *char vr[150]* : În această variabilă reținvremea pentru ziua respectivă ce va fi afișată tuturor clienților ce vor avea opțiunea pentru afișarea vremii.

- *char co[150]* : În această variabilă reținprețul combustibilului pentru ziua respectivă ce va fi afișată tuturor clienților ce vor avea opțiunea pentru afișarea prețului combustibilului.

Baza de date conține 5 tabele: *client*, *vreme*, *combustibil*, *sport*, *viteza*. Aceastea sunt descrise mai jos.

În tabelul *client* se află datele despre conturile clienților, în *sport*, *vreme* și *combustibil* se află informații necesare opțiunilor respective, iar în tabela *viteză* se află informații despre străzile din zonă și restricțiile necesare.

```
mysql> describe viteza;
+--------+---------------+------+-----+---------+----------------+
| Field  | Type          | Null | Key | Default | Extra          |
+--------+---------------+------+-----+---------+----------------+
| viteza | int           | YES  |     | NULL    |                |
| strada | varchar(1000) | YES  |     | NULL    |                |
| id     | int           | NO   | PRI | NULL    | auto_increment |
+--------+---------------+------+-----+---------+----------------+
3 rows in set (0,00 sec)

mysql> select * from viteza;
+--------+--------------------------+----+
| viteza | strada                   | id |
+--------+--------------------------+----+
|     30 | Strada Stefan Cel Mare   |  1 |
|     50 | Strada Mihai Eminescu    |  2 |
|     50 | Strada Calea Bucovinei   |  3 |
|     45 | Strada Calea Unirii      |  4 |
|     50 | Strada Piata Unirii      |  5 |
|     45 | Strada Ion Luca Caragiale |  6 |
|     50 | Strada Liviu Rebreanu    |  7 |
|     45 | Strada Ion Creanga       |  8 |
|     50 | Strada Mihail Sadoveanu  |  9 |
|     50 | Strada Ioan Slavici      | 10 |
|     50 | Strada Madalina Carausu  | 11 |
|     45 | Strada Muncii            | 12 |
|     45 | Strada Marasesti         | 13 |
|     40 | Strada Oituz             | 14 |
|     40 | Strada Eroilor           | 15 |
|     40 | Strada 1 mai             | 16 |
|     45 | Strada 1 decembrie       | 17 |
|     45 | Strada Ciprian Porumbescu | 18 |
|     50 | Strada Castanilor        | 19 |
|     50 | Strada Libertatii        | 20 |
+--------+--------------------------+----+
20 rows in set (0,00 sec)
```

```
mysql> describe client;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| nume       | varchar(100) | NO   |     | NULL    |       |
| prenume    | varchar(100) | NO   |     | NULL    |       |
| id         | int          | NO   | PRI | NULL    |       |
| viteza     | int          | YES  |     | NULL    |       |
| vreme      | varchar(1000)| YES  |     | NULL    |       |
| sport      | varchar(1000)| YES  |     | NULL    |       |
| combustibil| varchar(1000)| YES  |     | NULL    |       |
| strada     | varchar(1000)| YES  |     | NULL    |       |
| accident   | varchar(1000)| YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
9 rows in set (0,01 sec)

mysql> select * from client;
+---------+---------+----+--------+-------+-------+------------+-----------------------+----------+
| nume    | prenume | id | viteza | vreme | sport | combustibil | strada                | accident |
+---------+---------+----+--------+-------+-------+------------+-----------------------+----------+
| petrea  | daniela |  1 |     30 | 1     | 1     | 1          | Strada Mihai Eminescu | NULL     |
| haiura  | carla   |  2 |      5 | 1     | 1     | 0          | Strada 1 mai          | NULL     |
| haiura  | andreea | 12 |     10 | 0     | 1     | 1          | Strada Madalina Carausu | NULL   |
| haiura  | agnezia | 24 |     19 | 1     | 1     | 0          | Strada Libertatii     | NULL     |
| haiura  | tadeus  | 50 |      3 | 1     | 1     | 0          | Strada Calea Unirii   | NULL     |
| lazurca | dana    | 68 |     31 | 1     | 1     | 1          | Strada Castanilor     | NULL     |
| h       | h       | 80 |     13 | 1     | 0     | 1          | Strada Oituz          | NULL     |
+---------+---------+----+--------+-------+-------+------------+-----------------------+----------+
7 rows in set (0,00 sec)
```

```
mysql> describe vreme;
+-----------+---------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-----------+---------------+------+-----+---------+----------------+
| descriere | varchar(1000) | YES  |     | NULL    |                |
| id        | int           | NO   | PRI | NULL    | auto_increment |
+-----------+---------------+------+-----+---------+----------------+
2 rows in set (0,01 sec)

mysql> select * from vreme;
+-----------------------------------------------------+----+
| descriere                                           | id |
+-----------------------------------------------------+----+
| Ninge! Aveti grila la drum!                         |  1 |
| Se apropie o furtuna! Grija la drum!                |  2 |
| Este o zi insorita! O vreme perfecta pentru condus! |  3 |
+-----------------------------------------------------+----+
3 rows in set (0,00 sec)
```

```
mysql> describe sport;
+----------+---------------+------+-----+---------+----------------+
| Field    | Type          | Null | Key | Default | Extra          |
+----------+---------------+------+-----+---------+----------------+
| id       | int           | NO   | PRI | NULL    | auto_increment |
| descriere | varchar(1000) | YES  |     | NULL    |                |
+----------+---------------+------+-----+---------+----------------+
2 rows in set (0,00 sec)

mysql> select * from sport;
+----+-----------------------------------------------------------------------------------------------------------+
| id | descriere                                                                                                 |
+----+-----------------------------------------------------------------------------------------------------------+
|  1 | Va avea loc un meci de fotbal astazi de la 18:00 intre Steaua-Bucuresti si Rapid!                         |
|  2 | Maine la ora 14:30 va avea loc un meci de tenis intre Simona Halep si Serena Williams!                    |
|  3 | Scor nemaipomenit! 4-2 pentru Dinamo!                                                                     |
|  4 | Astazi va avea loc finala Campionatului National de Handbal intre CS Minaur Baia Mare si SCM Ramnicu Valcea! |
|  5 | Larisa Iordache si-a anuntat retragerea din gimnastica la 25 de ani!                                      |
+----+-----------------------------------------------------------------------------------------------------------+
5 rows in set (0,00 sec)
```

```
mysql> describe combustibil;
+-----------+---------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-----------+---------------+------+-----+---------+----------------+
| id        | int           | NO   | PRI | NULL    | auto_increment |
| descriere | varchar(1000) | YES  |     | NULL    |                |
+-----------+---------------+------+-----+---------+----------------+
2 rows in set (0,01 sec)

mysql> select * from combustibil;
+----+-------------------------------------------+
| id | descriere                                 |
+----+-------------------------------------------+
|  1 | Motorina=6.00, Benzina=5.92, iar GPL=3.72! |
|  2 | Motorina=5.92, Benzina=5.92, iar GPL=3.79! |
+----+-------------------------------------------+
2 rows in set (0,00 sec)
```

**În client există două while-uri**:

```
while(strcmp(msg, "Login efectuat cu succes!")!=0)
{...
}
while(1)
{
    bzero (msg, 200);
    if (read (sd, msg, 200) < 0)
    {
        perror ("[client]Eroare la read() de la server.\n");
        return errno;
    }
    if(strncmp(msg, "Nu mergeti cu viteza regulamentara", 30)==0)
    {
        printf ("\n[client]Mesajul primit este: %s\n", msg);
    }
    else
        if(strncmp(msg, "vreme", 5)==0)
        {
            printf ("\n[client]Mesajul primit este: %s\n", msg+6);
        }
        else
            if(strncmp(msg, "combustibil", 11)==0)
            {
                printf ("\n[client]Mesajul primit este: %s\n", msg+12);
            }
            else
                if(strncmp(msg, "sport", 5)==0)
                {
                    printf ("\n[client]Mesajul primit este: %s\n", msg+6);
                }
                else
                    if(strncmp(msg, "ATENTIE", 7)==0)
                    {
                        printf ("\n[client]Mesajul primit este: %s\n", msg);
                    }
                    else
```

Primul while conține partea în care clientul încearcă să se logheze. Până când serverul nu îi transmite mesajul *Login efectuat cu succes!*, acesta nu trece în a doua parte. Următorul while este infinit și va afișa mesajele transmise de server.

```
else
    if(strncmp(msg, "Aveti viteza mica de", 20)==0)
    {
        printf ("\n[client]Mesajul primit este: %s\n", msg);
        bzero (msg, 200);
        read (0, msg, 200);
        if(strncmp(msg, "exit", 4)==0)
        {
            if (write (sd, msg, 200) <= 0)
            {
                perror ("[client]Eroare la write() spre server.\n");
                return errno;
            }
            close(sd);
            exit(1);
        }
        else
        {
            if (write (sd, msg, 200) <= 0)
            {
               perror ("[client]Eroare la write() spre server.\n");
               return errno;
            }
            bzero (msg, 200);
            if (read (sd, msg, 200) < 0)
            {
                perror ("[client]Eroare la read() de la server.\n");
                return errno;
            }
            // afisam mesajul primit
            printf ("[client]Mesajul primit este: %s\n", msg);
        }
    }
```

Atunci care se trece la al doilea while, clientul primește și afișează informațiile trimise de server. Utilizatorul are dreptul de a tasta doar atunci când are viteza mai mică de *20 km/h* deoarece acesta nu are cum să părăsească aplicația dacă are viteza prea mare, din măsură de siguranță. Dacă viteza este mică, acesta poate să raporteze un accident sau să părăsească aplicație.

**În server există, la fel, doua while-uri**:

```
while(ok==0)
{ ...
}
avr=acces_la_vreme(id);
aco=acces_la_combustibil(id);
asp=acces_la_sport(id);
sleep(1);
while(1)
{
        srand(time(0));
        int viteza=rand()%70;
        printf("[server]Trimitem viteza catre baza de date...\n");
        int r=adauga_viteza_in_baza_de_date(id, viteza);
        if(r==-1)
           printf("[server]A aparut o eroare in baza de date!\n");
        else
            printf("[server]S-a adaugat viteza in baza de date!\n");
        int bytes;      // numarul de octeti cititi
        char msg[200];    //mesajul primit de la client
        int verif=verif_accident(id);
        if(verif==1) //exista accidente si trebuie sa le afisez caci nu eu l-am declarat
        {
            printf("Exista accident\n");
            strcpy(msg, mesaj_accident());
            if(trimite(msg, tdL.cl, strlen(msg))==0)
            {
                perror ("[server] Eroare la write() catre client.\n");
                return 0;
            }
        }
        else
        {
            if(verif==-3)    // -3 eu am raportat accidentul
            {
                printf("Eu am raportat accidentul asa ca il sterg\n");
                sterge_accident(id);
            }
```

Primul while verifică dacă clientul se poate loga cu succes. În cazul de login se folosește funcția de căutare, iar în cazul de autentificare se folosește și funcția ce adaugă informațiile clientului în baza de date. Nu se poate trece de primul while dacă serverul nu transmite mesajul *Login efectuat cu succes!* către client.

```
else
{
    int d=viteza_scazuta(id);
    if(d>0)
    {
        sprintf(msg, "Aveti viteza mica de %d km/h. Raportati un accident?", d);
        if(trimite(msg, tdL.cl, strlen(msg))==0)
        {
            perror ("[server] Eroare la write() catre client.\n");
            return 0;
        }
        bytes = read (tdL.cl, msg, 200);
        if (bytes < 0)
        {
            perror ("Eroare la read() de la client.\n");
            return 0;
        }
        if(strncmp(msg, "exit", 4)==0)
        {
            printf ("[server]S-a deconectat clientul cu descriptorul %d.\n",tdL.cl);
            close (tdL.cl);
            pthread_exit(NULL);
        }
        else
            if(strncmp(msg, "da", 2)==0)
            {
                strcpy(msg, "Vom anunta ceilalti participanti la trafic! Multumim pentru informatie!");
                if(trimite(msg, tdL.cl, strlen(msg))==0)
                {
                    perror ("[server] Eroare la write() catre client.\n");
                    return 0;
                }
                if(adauga_accident_in_baza_de_date(id)==-1)
                {
                    perror ("[server] Eroare la adauga_accident_in_baza_de_date().\n");
                    return 0;
                }
            }
            else
            {
                strcpy(msg, "Ne bucuram sa auzim asta! Drum bun in continuare!");
                if(trimite(msg, tdL.cl, strlen(msg))==0)
                {
                    perror ("[server] Eroare la write() catre client.\n");
                    return 0;
                }
            }
    }
```

Cel de-al doilea while ia viteza clientului și în funcție de aceasta verifică dacă este neregulamentară sau dacă este mică. În cazul vitezei scăzute, serverul va întreba clientul dacă există un accident, iar clientul are trei variante de răspuns: *exit, da, nu*. În cazul tastării primei comenzi, acesta va ieși din aplicație, în al doilea caz, va raporta un accident pe strada unde se află, iar îl ultimul caz va rămâne în aplicație fără să se modifice nimic.

**Funcția adauga_in_baza_de_date**:

```
int adauga_in_baza_de_date(char x[100])
{
    MYSQL *conn;
    MYSQL_ROW row;

    char *server = "localhost";
    char *user = "isabela";
    char *password =  ascuns
    char *database = "isabela";

    conn = mysql_init(NULL);


    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0))
    {
      printf("[server]Eroare la conectarea cu serverul %s din MySQL. Error: %s\n", server, mysql_error(conn));
      return -1;
    }
    char nume[50], prenume[50], nr[10];
    int id;
    int i=0;
    while(x[i]!='.'&&i<strlen(x))
    {
      nume[i]=x[i];
      i++;
    }
    nume[i]='\0';
    i++;
    int c=0;
    while(x[i]!='.'&&i<strlen(x))
    {
      prenume[c]=x[i];
      i++;
      c++;
    }
    prenume[c]='\0';
    i++;
    c=0;
    while(i<strlen(x))
    {
      nr[c]=x[i];
      i++;
      c++;
    }
    nr[c]='\0';
    id=atoi(nr);
    char sqls[200];
```

```
    sprintf(sqls, "insert into client(nume, prenume, id) values ('%s','%s',%d)", nume, prenume, id);
    if (mysql_query(conn, sqls))
    {
      printf("Failed to execute query. Error: %s\n", mysql_error(conn));
      return -1;
    }
    mysql_close(conn);
    return 1;
}
```

Această funcție arată cum comunica serverul cu baza de date din MySql. Conexiunea este creeată în primul if, iar dacă apare o problemă în ceea ce privește conectarea cu baza de date, funcția va returna -1. În continuare sunt luate numele, prenumele și id-ul clientului în trei variabile. Interogarea bazei de date se face prin intermediul variabilei sqls ce va conține interogarea. În cazul unei erori la interogare, funcția returnează -1, iar în cazul în care serverul a reușit să adauge datele în baza de date returnează 1.

**Funcția viteza_regulamentara**:

```
int viteza_regulamentara(int x)
{
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;

    char *server = "localhost";
    char *user = "isabela";
    char *password =  ascuns
    char *database = "isabela";

    int vit, vit_leg;
    conn = mysql_init(NULL);
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0))
    {
        printf("Failed to connect MySQL Server %s. Error: %s\n", server, mysql_error(conn));
        return 0;
    }

    char sql[300];

    sprintf(sql, "select viteza from client where id=%d", x);
    if (mysql_query(conn, sql))
    {
        printf("Failed to execute quesry. Error: %s\n", mysql_error(conn));
        return 0;
    }

    res = mysql_store_result(conn);
    if (res == NULL)
    {
        return 0;
    }

    row = mysql_fetch_row(res);
    vit=atoi(row[0]);

    if (mysql_query(conn, "select count(*) from viteza"))
    {
        printf("Failed to execute quesry. Error: %s\n", mysql_error(conn));
        return 0;
    }

    res = mysql_store_result(conn);
    if (res == NULL)
    {
        return 0;
    }
    row = mysql_fetch_row(res);
    int nr=atoi(row[0]); //nr de linii din tabel;
    srand(time(0));
    int id=1+rand()%nr;
```

Această funcție comunică cu tabela *viteza* și tabela *client* din baza de date. În variabila *vit* se reține viteza clientului, iar în variabila *vit_leg* se reține viteza legală de pe strada respectivă. În urma primei interogări se ia viteza clientului, iar prin cea de-a doua se află numărul de restricții din tabela *viteza*. Variabila *nr* reține acest număr, iar variabila *id* va lua un număr random între 1 și *nr*.

```
    sprintf(sql, "select strada from viteza where id=%d", id);
    if (mysql_query(conn, sql))
    {
        printf("Failed to execute quesry. Error: %s\n", mysql_error(conn));
        return 0;
    }

    res = mysql_store_result(conn);
    if (res == NULL)
    {
        return 0;
    }
    row = mysql_fetch_row(res);
    char st[100];
    strcpy(st, row[0]);

    sprintf(sql, "update client set strada='%s' where id=%d", st, x);
    if (mysql_query(conn, sql))
    {
        printf("Failed to execute quesry. Error: %s\n", mysql_error(conn));
        return 0;
    }

    sprintf(sql, "select * from viteza where id=%d", id);
    if (mysql_query(conn, sql))
    {
        printf("Failed to execute quesry. Error: %s\n", mysql_error(conn));
        return 0;
    }

    res = mysql_store_result(conn);
    if (res == NULL)
    {
        return 0;
    }
    row = mysql_fetch_row(res);
    vit_leg=atoi(row[0]);

    mysql_free_result(res);
    mysql_close(conn);
    if(vit<=vit_leg)
        return 1;
    else
        return vit_leg;
}
```

În continuare, următoarea interogare selectează strada din tabela *viteza* unde id-ul este reținut în variabila *id*. Strada va fi reținută în *st*. Se va adăuga în baza de date strada clientului cu id-ul furnizat la apel în variabila *x*. Ultima interogare selecteaza viteza din tabela *viteza* și va fi reținută în variabila *vit_leg*. Dacă *vit* este mai mică sau egală decât *vit_leg* atunci clientul merge regulamentar, iar în caz contrar, funcția returnează variabila *vit_leg* pentru a putea înștiința clientul cu privire la restricția de viteză.

## 5 Concluzii

La finalul proiectului am realizat o aplicație în care utilizatorii se pot conecta prin intermediul unui cont și în care pot alege dacă doresc să primească mesaje referitoare la vreme, sport și prețul combustibilului la stațiile peco.

Viteza cu care circulă fiecare utilizator va fi trimisă automat către server, iar această opțiune poate fi îmbunătățită astfel încât viteza să fie preluată direct prin intermediul unui GPS.

În aplicația pe care am implementat-o, mesajul de atenționare în cazul unui accident este trimis către toți utilizatorii activi, însă această opțiune poate fi îmbunătățită astfel încât, doar utilizatorii ce se află în apropierea străzii respective să primească mesajul, îmbunătățirea fiind posibilă tot prin intermediul unui GPS.

În concluzie, aplicația implementată funcționează conform cerințelor, iar clienții pot beneficia de monitorizarea traficului pe o anumită regiune.

## References

1. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
2. https://stackoverflow.com/
3. https://www.cyberciti.biz/tips/linux-unix-connect-mysql-c-api-program.html
4. https://linux.die.net/man/
5. https://app.diagrams.net/
6. https://www.geeksforgeeks.org/
7. https://man7.org/linux/man-pages/man7/pthreads.7.html