Peltea Bogdan

Monitorizarea traficului

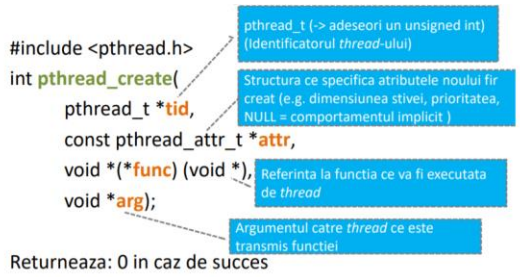
Descrierea proiectului:

Implementarea unui sistem care va fi capabil să gestioneze traficul și să ofere informații virtuale șoferilor. Șoferii vor avea posibilitatea să raporteze incidente din trafic spre sistem. Ulterior, aceste update-uri vor fi trimise către toți participanții din trafic. Fiecare mașina va trimite automat informații despre viteza cu care circula către sistem. Apoi, sistemul va atenționa fiecare șofer despre anumite restricții de viteza sau când exista un blocaj in trafic. De asemenea, fiecare șofer se va putea abona la diferite informații,cum ar fi:informații despre vreme, evenimente sportive, prețuri ale combustibililor la fiecare stație peco etc.

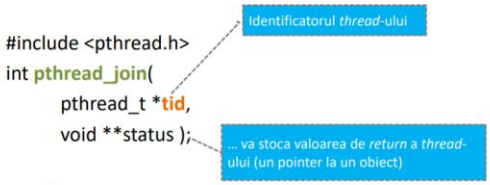
Pentru realizarea acestui sistem avem nevoie de:

1. Conexiune TCP folosind threads
2. O bază de date
3. O interfata grafică
4. O implementare GPS

1. Utilizăm o conexiune TCP folosind threads,deoarece ne dorim ca clienții noștri să primească toate informațiile date de server(ex: vreme,accidente),dar și de la client spre server. Folosim thread-uri pentru a avea o comunicare unică pentru fiecare client care se va conecta la server și pentru ca serverul nostru să execute mai multe lucruri in paralel. De exemplu,un program de navigare poate realiza minim două lucruri in paralel: aduce date din rețea și în paralel le afișează. Firele de execuție faciliteaza ca programarea să fie mai apropiata de gândirea umană. Omul,de obicei este ocupat cu mai multe lucruri in acelasi timp.Spre exemplu, un profesor in timp ce predă trebuie sa fie atent si la studenții care îl asculta.

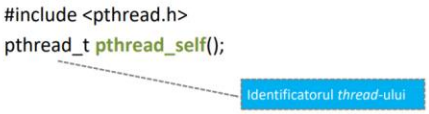


Returneaza: 0 in caz de succes

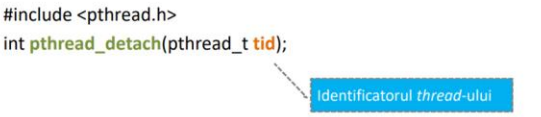


- Realizeaza asteptarea terminarii unui anumit *thread*

Returneaza: 0 in caz de succes



Returneaza: ID-ul *thread*-ului care a apelat primitiva



Thread-urile pot fi:

- *joinable*: cind thread-ul se termina, ID-ul si codul de iesire sunt pastrate pina cand se apeleaza pthread_join() <– comportament implicit
- *detached:* cand thread-ul se termina toate resursele sunt eliberate

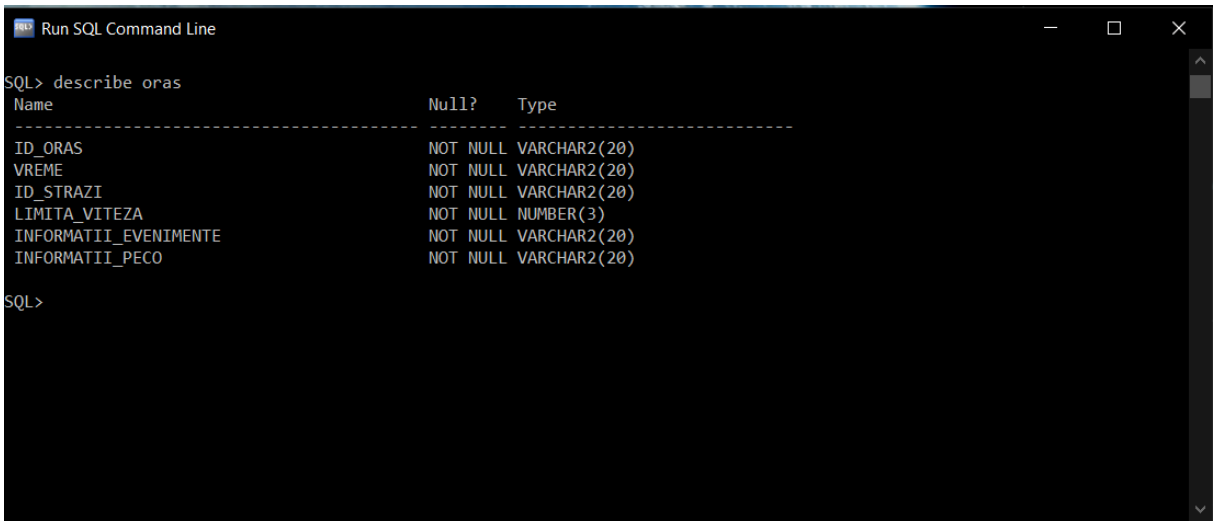
Returneaza: 0 in caz de succes

#include <pthread.h>
void pthread_exit(void*status);
- Terminarea unui *thread*

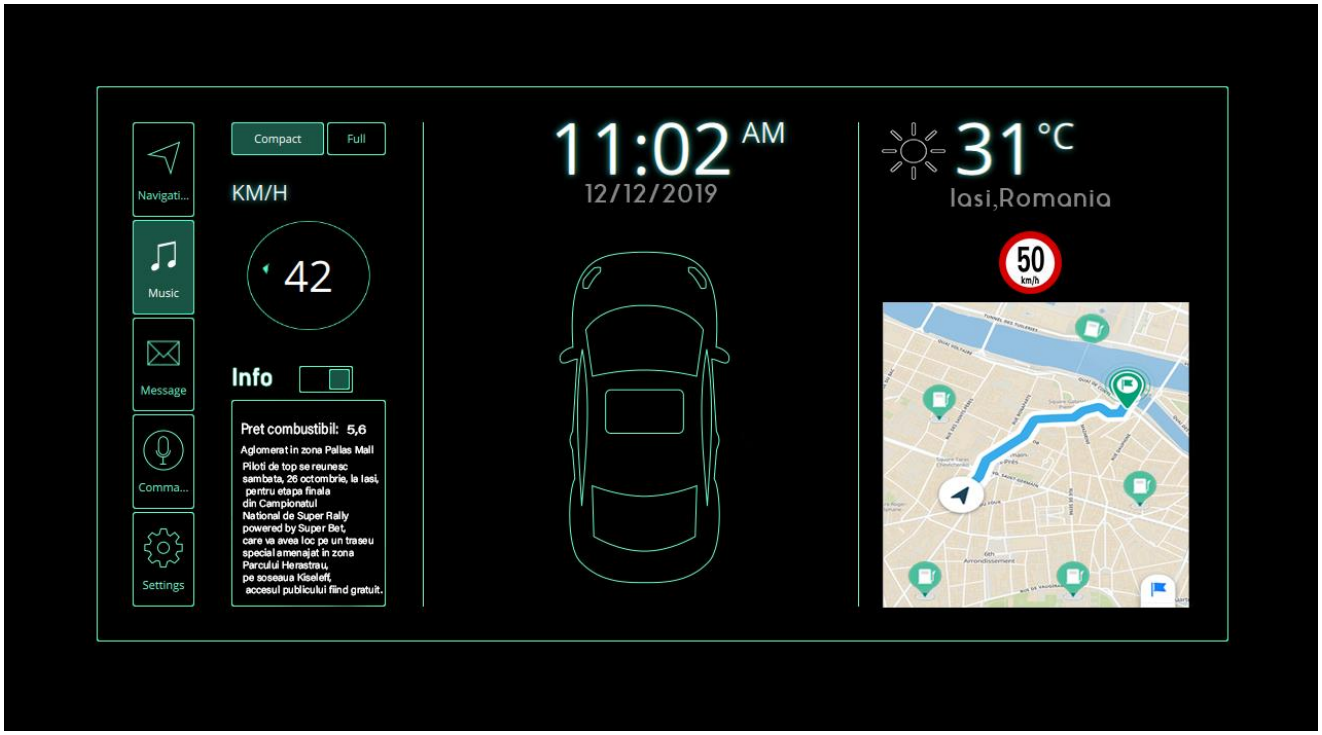
Thread-urile se pot termina:

- Functia executata de *thread* returneaza (Obs. Valoarea de return este void * si va reprezenta codul de iesire a *thread*-ului)
- Daca functia *main* a procesului returneaza sau oricare din *thread*-uri a apelat *exit*(), procesul se termina

2. Vom avea nevoie de o bază de date pentru a stoca date precum: climatul din fiecare oraș,evenimentele sportive,prețurile de la peco precum si restricțiile de viteza pe drumuri. Pentru aceasta vom folosi MySQL.



3. Un sistem bine pus la punct necesită o interfață grafică cât mai atractivă pentru client. Cu ajutorul programului Qt.



Imaginea de mai sus conține aspecte ce vor fi prezente doar clientului. Acesta va putea să se aboneze la informații în plus în funcție de locația în care se află.Pentru a simula viteza cu care circula automobilul ,clientul va putea modifica în timp real cu ajutorul discului (imaginea din stânga).Dacă viteza va depăși limita legală pe sectorul de drum pe care se deplasează automobilul, clientul va primi notificare de la server.Viteza cu care se deplasează automobilul va fi transmisă serverului o data la un minut urmând ca acesta să verifice în baza de date dacă viteza este cel mult egală cu viteza stocata in baza de date.

4. Implementarea unui GPS se va face cu ajutorul Qt Positioning(C++) care are o componantă Qt Location Api și este vorba despre poziția geografică și adresele unor locuri.Poziționarea conține o clasă QGeoCoordinate ,ce este compusă din latitudine,longitudine și altitudine în metri. QGeoLocation include QGeoCoordinate plus adrese și dimensiunea informațiilor. API-ul permite, de asemenea, dezvoltatorului să controleze și sursa informațiilor poziționale. Sursele de date despre locație sunt create prin subclase QGeoPositionInfoSource si furnizarea obiectelor QGeoPositionInfo prin semnalul QGeoPositionInfoSource :: positionUpdated(). Clientii care cer date despre locație se pot conecta prin clasa positionUpdated() și pot apela startUpdates() sau requestUpdate() pentru a declanșa distribuția datelor delocatie. Partajarea datelor de locație poate fi oprită apelând funcția stopUpdates().

```
class MyClass : public QObject
5.  {
6.      Q_OBJECT
7.  public:
8.      MyClass(QObject *parent = 0)
9.          : QObject(parent)
10.     {
11.         QGeoPositionInfoSource *source = QGeoPositionInfoSource::createDefaultSource(this);
12.         if (source) {
13.             connect(source, SIGNAL(positionUpdated(QGeoPositionInfo)),
14.                     this, SLOT(positionUpdated(QGeoPositionInfo)));
15.             source->startUpdates();
16.         }
17.     }
18.
19. private slots:
20.     void positionUpdated(const QGeoPositionInfo &info)
21.     {
22.         qDebug() << "Position updated:" << info;
23.     }
24. }
```

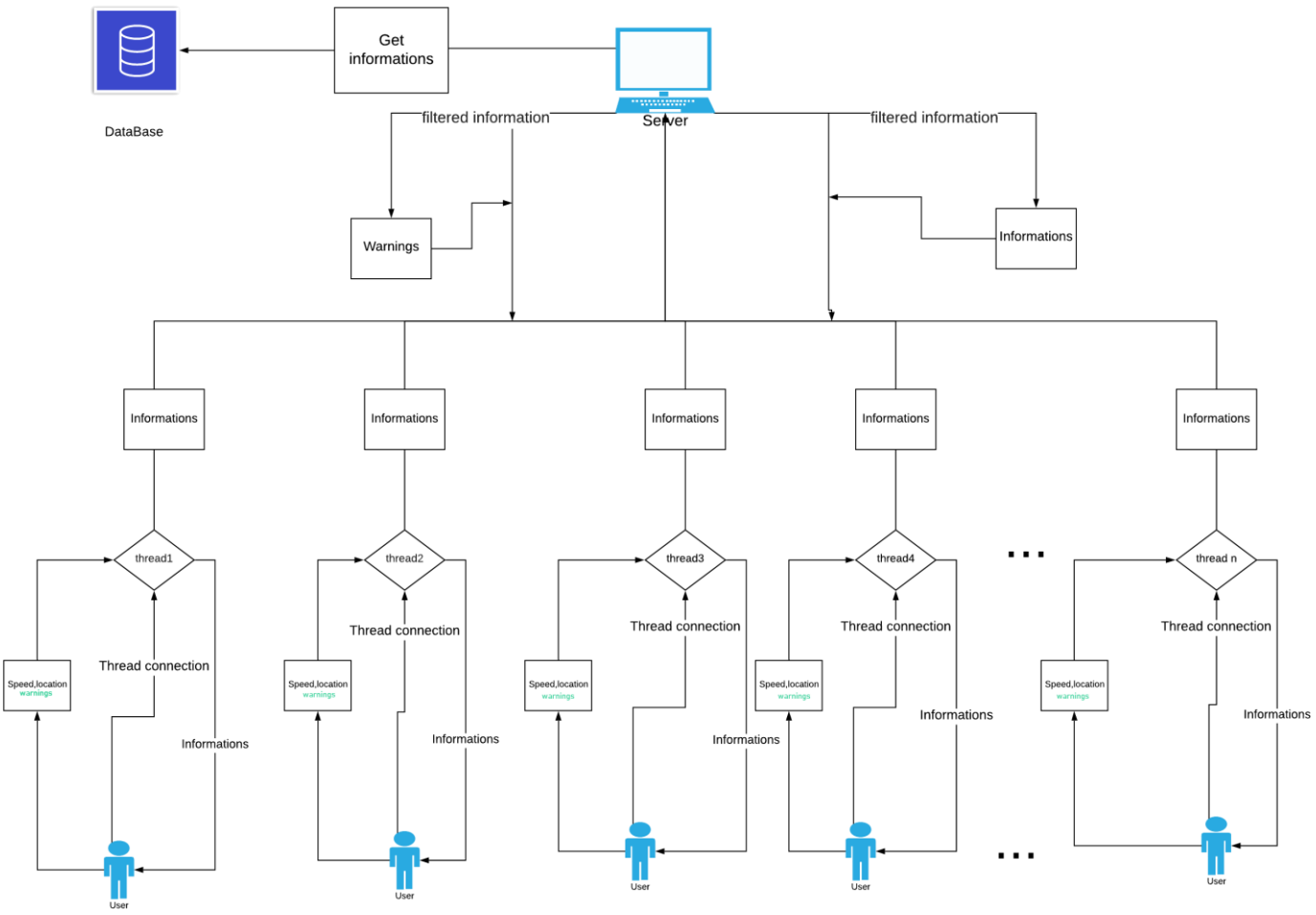


Diagrama de mai sus exemplifică procedeul prin care aplicația va funcționa. Pentru fiecare client va exista un proces unic cu care va comunica. O data la un minut fiecare client transmite viteza cu care se deplasează urmând ca aceasta informație să fie transmisă serverului după care acesta va verifica dacă viteza se încadrează în limitele impuse în baza de date la fel și poziția acestuia.Totodata acesta va putea transmite informații legate de accidente , dupa care, serverul va reține informația,o va filtra și o va transmite tuturor utilizatorilor. Informațiile în plus vor fi preluate din baza de date si transmise clienților care sunt abonați.

Concluzii: Am ales acest proiect deoarece reprezintă o provocare pentru mine,dar in același timp și o aplicație de care eu și alți șoferi avem nevoie și cel mai important lucru,simuleaza rularea aplicațiilor de tip navigație care au un foarte mare succes în zilele noastre și pe care mulți dintre noi o folosesc zi de zi. Ce aș îmbunătăți la aceasta aplicație ar fi implementarea unui gps real care poate monitoriza în permanență clientul și care acesta transmite informații serverului despre locația lui în timp real.Un alt lucru ar fi implementarea unei hărți care să fie bine pusă la punct și să conțină detalii precum:numele tuturor străzilor cât și limitele de viteză pe toate sectoarele de drum.

Documentatie:

https://doc.qt.io/qt-5/location-positioning-cpp.html

https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf

https://profs.info.uaic.ro/~bd/wiki/index.php/Pagina_principal%C4%83