# Introducere

----------------------------------------?????\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Transportul public reprezintă una din principalele modalități de deplasare în interiorul unei localități, punând la dispoziție publicului larg posibilitatea de a ajunge dintr-un loc în altul, fără a avea un mijloc de transport personal chiar și între distanțele mari din mediul urban. În același timp folosirea transportului public poate fi un factor important pentru reducerea cheltuielilor personale daca este folosit în schimbul autovehicolului. Alte beneficii ale transportului public ar fi: decongestionarea traficului, reducerea emisiilor de carbon, crearea de locuri de muncă și multe altele.

În cadrul acestei lucrări se urmărește dezvoltarea unui produs software pentru eficientizarea informării cu privire la toate informațiile necesare folosirii transportului în comun. Aplicația dorește să îmbunătățească formele actuale de informare printr-o structură cât mai dinamică și configurabilă pentru nevoile fiecărui utilizator.

## Punerea problemei

Observăm că transportul public nu ar fi benefic doar pentru noi din punct de vedere financiar cât și mediului înconjurător, totuși în utilizarea lui intervin două probleme esențiale:

* cunoașterea traseelor și stațiilor mijloacelor de transport;
* cunoașterea momentului când mijlocul de transport ajunge într-o anumită stație.

Pentru rezolvarea primei probleme și anume informarea populației cu privire la rutele mijloacelor de transport în comun, cea mai frecventă metodă folosită este aceea de a afișa în majoritatea stațiilor și în interiorul mijloacelor de transport, o hartă ce prezintă toate traseele și statiile din acel oraș.

A doua problema este rezolvată în mare parte ca și prima, fiind afisat un orar prestabilit cu intervalele de circulație sau in cel mai bun caz un afișaj cu informații în timp real despre momentul cand va ajunge mijlocul de transport în acea stație. Totuși chiar daca ar părea că nevoia de informare a populației ar fi îndeplinită, o întrebare ar fi: cât de repede se actualizează hărțile din stații din tot orașul la o modificare de traseu a unei linii? Daca ar fi o modificare temporară de traseu cât de accesibilă va fi aceasta informație? Sau cum putem folosi transportul în comun daca nu știm unde sunt stațiile sau care ar fi cea mai apropiata stație de noi?

Observăm că doar aceste căi de informare aduc după sine îngreunarea folosirii transportului în comun prin incertitudinea unor informații și folosirea ineficientă a transportului prin asteptarea nedorită în stație.

# Analiza stadiului actual în domeniul problemei

Pentru a intra în profunzimea problemei trebuie sa ne raportăm la un caz real și anume orașul Timișoara. Orașul Timișoara dispune de tot ceea ce a fost descris mai sus cât și de mijloace suplimentare de informare. Site-ul Regiei Autonome de Transport Timișoara (RATT) conține o harta a transportului cu autobuze: urban, expres și metropolitan fără trasee deviate temporar, existând la momentul de față trasee neactualizate, mai conține o hartă a transportului electric tot fara trasee derivate temporar și nu în ultimul rând conține informații cu privire la timpul în care va ajunge un mijloc de transport într-o stație. Pe lânga aceasta mai există o aplicație mobilă android care ne oferă un acces mai ușor la informațiile afișate pe pagina oficială R.A.T.T. și anume prea datele cu privire la momentul cand ajunge un mijloc de transport în stație si le afițează sub forma unei liste.

Totuși pentru folosirea cea mai eficientă a transportului public în Timișoara ar fi nevoie de o sursa de încredere de informare, cu date actualizate în mod continu, să poată fi accesată din orice locație ne-am afla și în care să găsim atât traseul actual al unei linii cât și timpii de sosire a mijloacelor de transport în comun pentru fiecare stație.

O soluție ar fi o aplicație mobilă ce ar avea acces întotdeauna la stațiile și traseele oficiale ale transportului în comun cu un mod interactiv de afisare pe o hartă dinamică a Timisoarei și pentru fiecare stație timpul de sosire al fiecărui mijloc de transport.

## Studiu asupra soluțiilor existente

La momentul actual există trei aplicații mobile pentru informarea cu privire la timpul de calatorie al mijloacelor de transport în comun, fiecare fiind dezvoltată pe cate una din principalele sisteme de operare mobile: iOS, Windows Phone și Android. În următoarele rânduri ne vom opri la aplicația folosită în Timișoara de cel mai mare numar de utilizatori, și anume Public Transport Timisoara pentru Android.

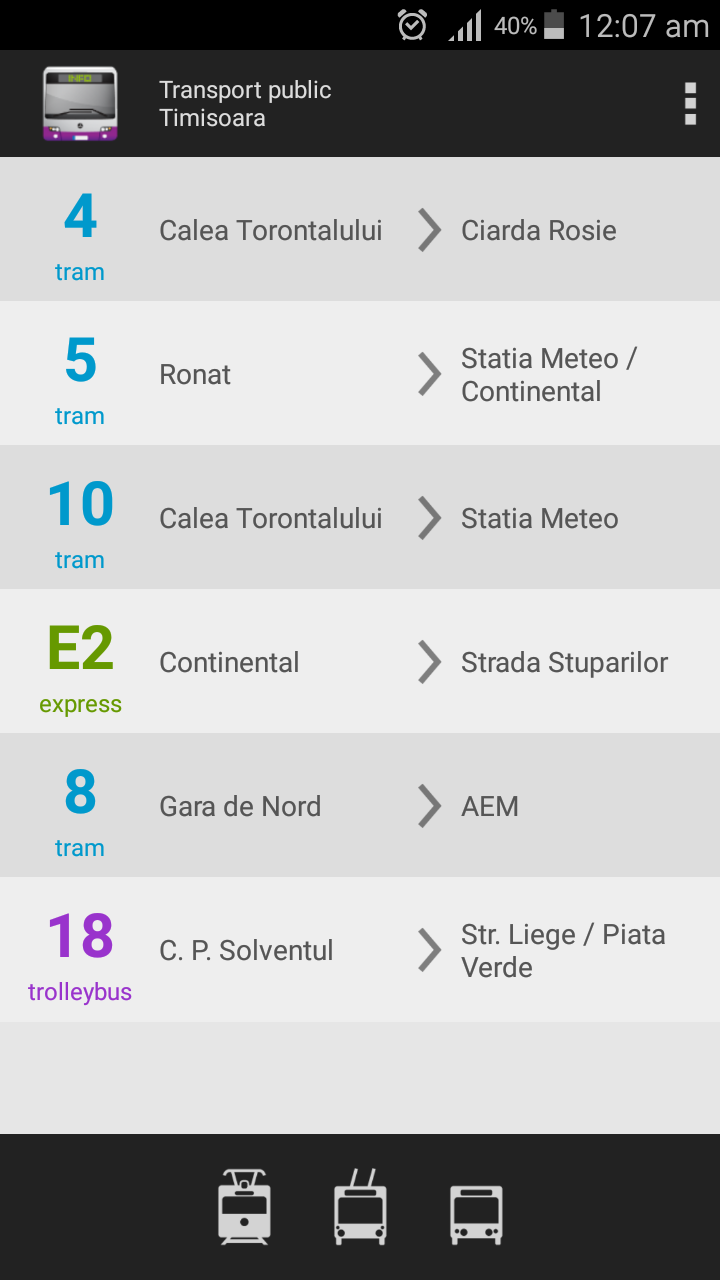


Figura . Aplicația „Public Transport Timișoara”

Aplicația dispune de o listă cu toate mijloacele de transport, iar la selectarea uneia dintre linii se va afișa o lista cu stațiile într-un anumit sens de deplasare cu timpii la care transportul ajunge în stație. Informațiile cu privire la timpul sosirii le extrage din calea:

<http://ratt.ro/txt/afis_msg.php?id_traseu=XXXX&id_statie=XXXX>

La momentul selectării unei anumite linii, aplicația va apela serverul RATT pentru fiecare stație cu parametrii trimiși: id-ul traseului selectat și id-ul stației. Stațiile și traseele disponibile sunt stocate în memoria telefonului printr-un fișier de tipul CSV creat manual cu ajutorul de mai multe persoane din mediul online, acesta putând fi actualizat doar prin actualizarea aplicației.

Prin folosirea îndelungată a acestei aplicații utilizatorii au observant că la orele de vârf, datorita numărului mare ale cererilor la server pentru fiecare stație în parte și a mulțimii de utilizatori, serverul nu mai poate face față, iar informația devine inaccesibilă în momentele critice ale zilei.

# Bazele teoretice. Tehnologi folosite

Pentru implementarea proectului prezentat în această lucrare s-au folosit o varietate de sisteme, platforme și unelte software, cum ar fi: sistemul de operare Android pe care s-a dezvoltat aplicația, mediul de programare Android Studio în care s-a implementat aplicația, API-urile Google Maps folosite la afișarea harților și baza de date SQLite pentru stocarea locală a stațiilor.

## Android OS

Android este un sistem de operare, având la bază ca nucleu Linux și este dezvoltat în mare parte pentru telefoanele mobile și tablete. Începând din anul 2013 Android este sistemul de operare cel mai raspândit pe dispozitivele mobile din întreaga lume.

Android s-a format în anul 2005, moment în care telefoanele mobile inteligente erau într-un moment de început din punct de vedere tehnologic, din acest motiv dezvoltarea telefoanelor inteligente la acel moment era o mină de aur.

Android, Inc a început ca o companie de software de sine stătătoare care nu o vândut nici un produs timp de doi ani. Cu o echipa mica de ingineri software ei plănuiau să construiască noua generație de software pentru telefoanele mobile inteligente. Concentrându-se pe ideea de a conecta utilizatorul cât mai mult cu internetul și de a crea un mediu propice pentru orice dezvoltator software, compania o avut un plan bine pus la bază, încât în anul 2005 cand softwere-ul a fost terminat, o mulțime de investitori erau interesați de noua generație a experienței mobile. În acel moment Google căuta o companie de telefoane mobile pentru a concura împotrivă la Microsoft și Blackberry. Fondatorii Google dorind ca mai multe telefoane mobile să dețină în mod implicit ca motorul lor de căutare, platforma deschisă Android le-a oferit exact ceea ce aveau ei nevoie.

Primul telefon Android a fost T-Mobile G1, un telefon cu o tastatură QWERTY și un ecran tactil de dimensiuni mari pentru vremea respectivă. Acesta a fost unul dintre produsele create sub considerentele Google și produse de HTC.



Figura . Primul telefon cu platforma Android

În momentul lansării telefonului era o perioadă în care publicul nu era obisnuit cu o interfață doar tactilă a telefonului, ci erau oameni care încă simțeau nevoia unei tastaturi cu butoane fizice, acest telefon îmbinăndu-le pe ambele.

De atunci Android a evoluat tot mai mult pastrăndu-și principiile cu care a început: de a deveni un produs deschis pentru modificări și accesibil gratuit tuturor, revoluționând cu totul gradul libertate și de conectivitate ce îl conferă utilizatorului unui telefon mobil. Android și-a extins aria de dispozitive pe care poate rula de la telefoane, tablete pana la ceasuri și dispozitive din interiorul unei mașini.

Sistemul de operare Android este o stivă de componente software care sunt împărțite în cinci mari straturi după cum se poate observa în diagrama de mai jos.



Figura . Straturile componentelor software ale sistemului de operare Android

*Application framework* reprezintă stratul cel mai înalt din arhitectura sistemului și este folosit de dezvoltatorii de aplicații în cea mai mare parte. Accesul la hardware din acest nivel se face cu ajutorul unor interfețe de programare definite în stratul de abstractizare hardware (HAL).

*Binder Inter-Process Comunication(IPC)* este un mecanism ce permite stratului superior să treacă anumite granițe ale proceselor și să apeleze anumit cod din stratul inferior de servicii Android. La nivelul de aplicație aceasta comunicație este ascunsă față de dezvoltator.

*Android System Services* reprezintă stratul de servicii și este compus într-o formă modulară. Android include doua grupe de servicii: de sistem, ce cuprinde servicii precum Window Manager, Notification Manager și servicii media ce includ servicii de capturare sau redare media.

*Hardware abstraction layer (HAL)* definește o interfață standard pentru componentele hardware. HAL permite implementarea funcționalităților fără a afecta sau modifica stratul superior al sistemului. Implementarea HAL este păstrată în fișiere de tipul .so și vor fi încărcate de sistem la momentul potrivit.

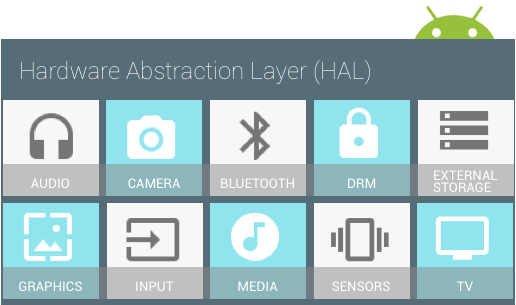


Figura . Stratul de abstractizare hardware

## **Android Studio**

Android Studio este platforma oficială de dezvoltare a aplicațiilor mobile Android, construită având la baza mediul de dezvoltare IntelliJ IDE creat de JetBrains. Este distribuită de către Google sub licența Apache Licence 2.0 ce oferă utilizatorilor dreptul gratuit de a folosi platforma gratuit atât în scopuri personale cât și în cele comerciale.

Fiecare proiect din Android Studio conține unul sau mai multe module ce conține urmatoarele foldere:

* *manifests*: conține fișierul AndroidManifest.xml ce prezintă informații esențiale despre aplicație, informații care trebuie să fie cunoscute de către sistem înainte de a rula codul. În conținutul fișierului este precizat numele pachetului Java al aplicației, aici se declara ce fel de permisiuni are nevoie aplicația pentru accesul parților protejate a unor diferite funcții API sau legături între alte aplicații, este descrisă lista de componente din aplicație: activități, servicii și în ce fel de condiții se rulează fiecare și lista de librarii folosite;
* *Java*: conține fișiere de cod sursă Java;
* *res*: conține fișiere non-cod cum ar fi scheme XML (layouts), anumite texte predefinite si imagini bitmap.

Mediul Android Studio folosește Gradle ca și fundație pentru sistemul de compilare, doar cu mai multe caracteristici configurate special pentru Android. Sistemul de compilare rulează ca o componentă integrată în meniul Android Studio și permite personalizarea, configurarea si extinderea procesului de compilare, crearea de multiple fișiere APK pentru aplicație, cu diferite caracteristici folosind acelasi proiect și aceleași module, de exemplu crearea de versiuni separate ale aplicației: o versiune gratuită si una platită. Fișierele de compilare Android Studio se numesc build.gradle și sunt fișiere text ce folosesc sintaxa Groovy pentru configurarea compilării. La importarea unu proiect existent, Android Studio genereaza automat fișierele necesare pentru compilare.

Android Studio asista la scrierea codului încât dispune de un dispozitiv de analiză de performanță a codului pe masură ce este scris. În modul de depanare ne ofera informații la fiecare linie cu privire la: valorile variabilelor, obiectele referențiate, valorile returnate de către metode. Tot în ajutorul dezvoltatorului, în vederea unei depanari și monitorizări cât mai exacte a aplicației, Android Studio dispune de informații cu privire la memorie si folosirea procesorului, gasirea obiectelor dezalocate, pierderilor de memorie, optimizarea performanței grafica si analizarea traficului internet.

Pentru a rula aplicația avem două opțiuni: folosind Android Emulator sau prin conectarea prin USB a unui dispozitiv cu un sistem de operare Android. Dupa selectarea uneia dintre opțiuni, se va instala aplicația pe emulator sau dispozitivul conectat, urmând ca la a doua rulare sa nu mai fie instalat din nou ci doar actualizat cu noile modificări efectuate, aceasta reducând semnificativ timpul de lansare pe dispozitiv sau emulator.

## Google Maps Android API

Google Maps Android API permite dezvoltatorilor să înglobeze harți Google în orice aplicație. Interfața manipulează în mod automat accesul la serverele Google Maps prin descarcărcarea datelor necesare, afișarea hărții și raspunderea la schimbarea sau apropierea unei anumite locații pentru furnizarea unor detalii suplimentare. Deasemenea permite adaugarea de diferite componente optice pentru îmbunatățirea experienției utilizatorului cum ar fi: adaugarea anumitor puncte semnificative pe hartă (eng. marker), desenarea a diferitor linii sau poligoane pe harta și alte straturi grafice suprapuse peste harta de bază oferită de interfață.

Pentru a accesa interfața, aplicația are nevoie de o cheie API pentru a i se permite accesul la serverele Google Maps. Acest cod este generat gratuit și nu limitează numarul de utilizatori al aplicației.

În mod implicit, fișierul XML ce definește locul unde va fi poziționată harta în aplicație, res/layout/activity\_maps.xml, va conține următorul cod:

<**fragment xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:map="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/map"  
 tools:context=".MapsActivity"  
 android:name="com.google.android.gms.maps.SupportMapFragment"** />

Fișierul Java, MapsActivity.java va implementa interfața OnMapReadyCallback care necesită suprascrierea a doua metode: onCreate si onMapReady. Prima metodă va fi apelată la pronirea aplicației și va porni procesul asincron de descărcare a hărților, iar a doua metodă se va executa atunci când hărțile s-au descărcat cu succes și cuprinde configurațiile de inceput ale harții cum ar fi: poziția de început a harții definită prin latitudine si longitudine, valoarea de apropiere față de coordonatele stabilite, metode pentru implementarea evenimentelor de selectare ale unor elemente grafice de pe hartă.

## Baza de date SQLite

 SQLite este „inima” ce stă la baza platformei Android ca suport pentru stocarea informațiilor într-o bază de date. Aceasta bază de date a fost dezvoltată în special pentru sisteme încorporate care dispun de o memorie scazută și putere de procesare a procesorului mică. Acest sistem nu este folosit doar de Android ci și de sistemul de operare Apple, iOS, sistemul de operare produs de BlackBerry cât și de alte platforme de acest gen. SQLite implementează un motor de baze de date SQL printr-o bibliotecă C de dimensiuni mici care nu necesită nici o configurare.

Motivele pentru care SQLite este prezent în numeroase sisteme încorporate și în lumea telefoanelor mobile sunt:

* Consumul de memorie scăzut;
* Ușurința de folosire;
* Disponibilitatea gratuită.

În timp ce consumul de memorie al bazei de date SQLite începe de la 50 kilobytes, rămâne tot la valori scăzute chiar și la proiectele mai mari cu o structura de date complexă, ajungând la o valoare de maxim câteva sute de kilobytes. În lumea mobilă memoria per proces, la fel ca și memoria totală folosită, este limitată față de sistemele desktop, de aceea SQLite este perfect pentru astfel de sisteme neadăugând prea mult la consumul de memorie totală al aplicației.

Ușurința în folosire reiese din faptul că SQLite nu rulează deloc nici un proces. SQLite se foloseste asemănător unei biblioteci, ajutând la accesarea fișierelor bazei de date. Nu are nevoie de nici o configurare sub nici o formă. Nu necesită configurarea portului, fără adăugarea de utilizatori, fara nici o gestionare a nivelelor de acces, nu are nevoie de nici o configurare a spațiului pentru tabele și multe altele. Tot conținutul bazei de date se va păstra într-un fișier și va fi creat la momentul inițializării bazei de date.

Codul SQLite este distribuit gratuit pentru domeniul public. Oricine este acceptat ca sa copieze, să modifice, să publice, să folosească, să vândă sau să distribuie codul original SQLite în versiune compilată sau nu, pentru orice scop, comercial sau necomercial și prin orice mijloace. Având în spate companii mari ce folosesc acest sistem, cum ar fi Google, Adobe sau Mozilla, proiectul este menținut activ, deci este de asteptat ca periodic să apară anumite îmbunătățiri și optimizăriale codului.

SQLite oferă mai puține tipuri de date decât într-o bază de date SQL normală, dar daca se folosesc alte tipuri de date, cum ar fi varchar, în comanda CREATE TABLE, atunci SQLite o sa le mapeze la un tip de date cât mai apropiat cu putință. În tabelul urmator sunt prezentate toate tipurile suportate de SQLite.

|  |  |
| --- | --- |
| Tip de date | Semnificație |
| NULL | Valoarea de nul |
| INTEGER | Orice număr întreg |
| REAL | Numere reale cu virgulă (8-Byte IEEE 754) |
| TEXT | Orice șir de caractere sau un singur caracter(UTF-8, UTF-16BE sau UTF-16LE) |
| BLOB | Colectie de date binare stocate sub un singur bloc |

Tabel . Tipurile de date folosite de SQLite

Cea mai mare problemă este lipsa unui tip de date pentru timp și a tipului boolean care va trebui reprezentat sub formă de întreg cu valori 0 sau 1, iar pentru a memora date calendaristice o posibilă implementare ar fi memorarea sub formă de șir de caractere în formatul ISO 8601. Reprezentând data de 27 iunie 2016 ar arăta sub forma „2016-06-27” sau dacă se memorează si ora „2016-06-27T09:30”.

După cum s-a menționat mai sus o baza de date SQLite este un simplu fișier accesat prin SQLite API. În Android aceste fișiere sunt stocate în mod implicit în directorul:

/data/data/<numele-pachetului>/databases

De exemplu pachetul aplicației prezentate în această lucrare este „ro.ratt.transport”, iar numele bazei de date este „transport.db”, deci fișierul bazei de date va fi:

/data/data/ro.ratt.transport/databases/transport.db

Deși fișierul bazei de date nu este stocat într-o locație accesibilă utilizatorului, fiind memorat în directorul privat al aplicației, orice informație sensibilă ar trebui criptată fără sa ne bazam pe faptul ca fișierul nu este la vedere. Aplicații cum ar fi SQLite Editor ar putea cu ușurința să citeasca orice informație memorata în baza de date daca nu ar fi criptată.

## Tehnologia JAVA

 Java este un limbaj de programare orientat pe obiecte, dezvoltat de către James Gosling în cadrul companiei Sun Microsystems și lansat în anul 1995. Limbajul a fost creat cu devizia pentru un program „scris o singură dată, rulat pe orice sistem”, însemnând că odată compilat, codul Java, va rula pe orice platformă ce suportă Java, fără a avea nevoie de recompilare. Java este folosit ca bază pentru o multitudine de aplicații pentru sisteme incorporate sau mobile, jocuri video, aplicații web și multe altele. Java a fost proiectat pentru dezvoltarea de aplicații portabile, pentru o performanță crescută a celor mai variate platforme posibile. Prin faptul că aplicațiile vor fi disponibile în cele mai rare medii, companiile pot furniza mai multe servicii prin creșterea productivității, a comunicațiilor, a colaborărilor și nu în ultimul rând a reducerii drastice a costurilor de deținere a unor aplicații pe mai multe platforme. Java a devenit neprețuită pentru dezvoltatori software din următoarele motive[[1]](#footnote-1):

* Scrierea de software pe o anumită platformă și rularea virtuală pe orice altă platformă;
* Crearea de programwe ce pot rula în interioul browser-ului web și accesarea serviciilor web disponibile;
* Combinarea de aplicații sau servicii utilizând limbajul Java pentru a crea posibilitatea unei personalizări de înaltă clasă a aplicațiilor și serviciilor;
* Scrierea de aplicații eficiente pentru telefoane mobile, microcontrolere, module wireless, senzori și orice alt dispozitiv electronic.

Java este cheia principală în Android deși sistemul de operare construit pe nucleul de Linux care este scris în întregime în C, Android folosește limbajul Java ca bază pentru aplicațiile Android. Cu toate acestea, Android nu folosește mașina virtuală standard Java, ci folosește codul Java ca un pas intermediar care va fi transformat în cod Dalvik. În funcție de versiunea Android, acesta poate fi interpretat ori de mașina virtuală Dalvik, ori compilat de Android Runtime în cod nativ. Android deasemenea nu furnizează toată biblioteca standard Java SE, cu toate că Android cuprinde biblioteci de clase ce includ o implementare independentă a unei mari parti din cele neintegrate. Acest lucru a condus la o dispută între Oracle și Google. Pe data de 7 mai 2012, judecătoria San Francisco a găsit că în cazul în care API-urile dețin drepturi de autor, atunci Google a vătămat drepturile de autor Oracle prin folosirea lor în dispozitive Android[[2]](#footnote-2).

# Soluția propusă. Implementare

## Flow Software. Organigrama sistemului.

### Sistemul de localizare a mijloacelor de transport

Pentru o bună gestionare a mijloacelor de transport, fiecare vehicul este echipat cu un dispozitiv de localizare GPS care este conectat la o rețea locala pentru centralizarea datelor. Odată ce informațiile privind localizarea mijloacelor de transport ajung la server, un algoritm de aproximarea timpilor va calcula pentru fiecare stație durata în care mijlocul de transport în comun va ajunge în respectiva stație. După terminarea calculelor, datele vor fi afișate sub forma de tabel în pagina web oficială a regiei de transport și pe panourile digitale plasate în stațiile din oraș.

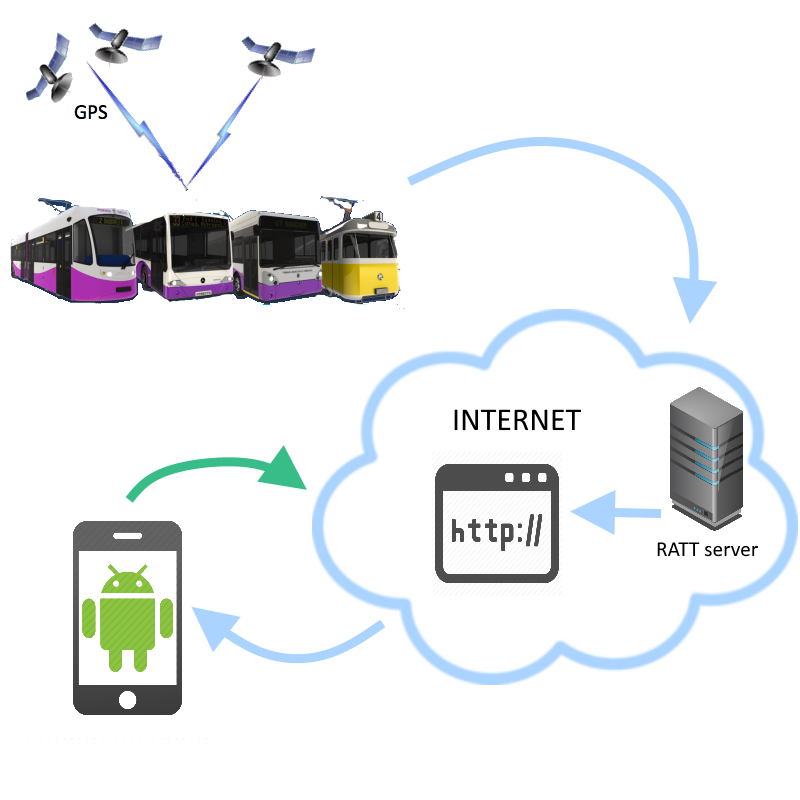
O reprezentare grafică a sistemului de localizare se gasește în figura de mai jos. 

Figura . Sistemul de localizare a transportului în comun

## Baza de date

Baza de date este folosită de aplicație pentru a memora toate stațiile și mijloacele de transport ce trec prin ele, împreună cu coordonatele geografice pentru amplasarea pe hartă. Baza de date se va crea și se va popula cu informație doar la prima rulare a aplicației urmând ca actualizarea ei să se realizeze printr-o actualizare a întregii aplicații. Folosind baza de date SQLite prezentă în Android, nu mai avem nevoie de nici o configurare ci doar definirea pentru numele fișierului bazei de date, a versiunii și apoi structura ei. Structura bazei de date se va descrie în tabelul următor:

|  |  |  |
| --- | --- | --- |
| Numele câmpului | Tipul | Descrierea informației |
| nr\_crt | INTEGER | Numărul curent al înregistrării, cheia primară a tabelei |
| lineID | INTEGER | Identificatorul numeric al liniei de transport |
| lineName | TEXT | Numele liniei de transport |
| stationID | INTEGER | Identificatorul numeric al stației de transport |
| rawStationName | TEXT | Numele stației în formatul publicat de ratt.ro |
| friendlyStationName | TEXT | Numele stației într-un format inteligibil |
| shortStationName | TEXT | Numele stației într-un format inteligibil prescurtat |
| junctionName | TEXT | Numele joncțiunii |
| lat | REAL | Coordonata latitudine a stației |
| lng | REAL | Coordonata longitudine a stației |
| invalid | TEXT | Dezactivarea stației daca nu mai este folosită |
| verificationDate | TEXT | Ultima actualizare a cămpului |
| route | TEXT | Directia de sens a liniei |

Tabel . Structura tabelei „Junctions”

Pentru manipularea bazei de date s-a creat clasa DBHandler ce va extinde SQLiteOpenHelper, o clasă abstractă prezentă în bibliotecile Android care ne va ajuta la crearea și gestionarea versiunilor bazei de date. DBHandler va suprascrie metodele onCreate și onUpgrade ce vor implementa codul pentru crearea tabelei resprectiv pentru actualizarea bazei de date.

@Override  
**public void** onCreate(SQLiteDatabase db) {  
 String query = **"CREATE TABLE "** + ***TABLE\_JUNCTIONS*** + **" ( '"** +  
 ***COLUMN\_INDEX*** + **"' INTEGER PRIMARY KEY, '"** +  
 ***COLUMN\_lineID*** + **"' INTEGER, '"** +  
 ***COLUMN\_lineName*** + **"' TEXT, '"** +  
 ***COLUMN\_stationID*** + **"' INTEGER, '"** +  
 ***COLUMN\_rawStationName*** + **"' TEXT, '"** +  
 ***COLUMN\_friendlyStationName*** + **"' TEXT, '"** +  
 ***COLUMN\_shortStationName*** + **"' TEXT, '"** +  
 ***COLUMN\_junctionName*** + **"' TEXT, '"** +  
 ***COLUMN\_lat*** + **"' REAL, '"** +  
 ***COLUMN\_lng*** + **"' REAL, '"** +  
 ***COLUMN\_invalid*** + **"' TEXT, '"** +  
 ***COLUMN\_verificationDate*** + **"' TEXT, '"** +  
 ***COLUMN\_route*** + **"' TEXT "** +  
 **");"**;  
 db.execSQL(query);  
}  
  
@Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
 db.execSQL(**"DROP TABLE IF EXISTS "** + ***TABLE\_JUNCTIONS***);  
 onCreate(db);  
}

Inserarea datelor în tabelă se face cu ajutorul metodei addJunction care primește ca parametru un obiect de tip Junction care la rândul lui conține toate campurile cuprinse de tabelă.

**public void** addJunction(Junction junction){  
 ContentValues values = **new** ContentValues();  
 SQLiteDatabase db = getWritableDatabase();  
 values.put(***COLUMN\_INDEX***, junction.getIndex());  
 values.put(***COLUMN\_lineID***, junction.getLineID());  
 values.put(***COLUMN\_lineName***, junction.getLineName());  
 values.put(***COLUMN\_stationID***, junction.getStationID());  
 values.put(***COLUMN\_rawStationName***, junction.getRawStationName());  
 values.put(***COLUMN\_friendlyStationName***, junction.getFriendlyStationName());  
 values.put(***COLUMN\_shortStationName***, junction.getShortStationName());  
 values.put(***COLUMN\_junctionName***, junction.getJunctionName());  
 values.put(***COLUMN\_lat***, junction.getLat());  
 values.put(***COLUMN\_lng***, junction.getLng());  
 values.put(***COLUMN\_invalid***, junction.isInvalid());  
 values.put(***COLUMN\_verificationDate***, junction.getVerificationDate());  
 values.put(***COLUMN\_route***, junction.getRoute());  
 db.insert(***TABLE\_JUNCTIONS***, **null**, values);  
 db.close();  
}

Clasa DBHandler mai cuprinde metode de returnare a valorilor înregistrate în baza de date, cum ar fi:

* **public** List<String> getListOfTransport(**int** option) ce returnează o listă cu mijloacele de transport de un anumit tip trimis ca parametru prin variabila option (1 - tramvai, 2 - trolebuz, 3 - expres, 4 - bus);
* **public** List<Station> getListOfStations(String line, String route) ce returnează lista stațiilor unei linii pe o anumită rută.

Așa cum s-a menționat și mai sus, înregistrările bazei de date se vor insera la prima rulare a aplicației, dintr-un fișier de tipul CSV (Comma-separated values). Acest tip de fișier stochează informație tabelară delimitând câmpurile înregistrării prin virgulă, iar înregistrarea prin sfărșitul de linie[[3]](#footnote-3). Un exemplu din fișier, deschis cu utilitarul Microsoft Excel, se poate observa în figura de mai jos:

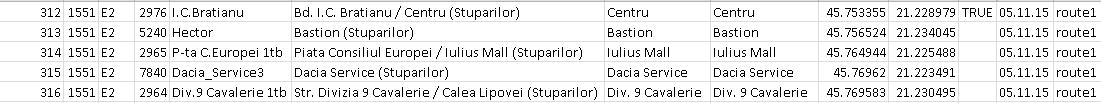


Figura . Conținutul fișierului de intrare

Fișierul va fi citit linie cu linie, cu ajutorul unu obiect de tip CSVReader preluat din biblioteca OpenCSV, creând o lista cu obiecte de tip Junction urmând ca apoi lista să fie adăugată element cu element în baza de date.

## Descărcarea timpilor de ajungere în stație

Informația cu privire la momentul de sosire al mijlocului de transport în stație, se va primi în două situații: atunci când linia va fi selectată pentru afișare sau dacă linia este afișată și butonul de actualizare este apăsat. Aceste informații vor fi preluate de pe serverul regiei de transport accesând următoarea locație, adăugând parametrul GET param1 cu valoarea numărului de indentificare a liniei dorită: <http://86.122.170.105:61978/html/timpi/sens0.php?param1=>.

În această pagină se vor afișa două tabele, primul tabel cu informații despre timpii de călatorie pentru traseul dus, iar al doilea informații pentru ruta de întoarcere a mijlocului de transport selectat. Tabelele vor conține urmatoarele câmpuri: numele liniei, numele stației și timpul de sosire a transportului în stație.

Procesul de preluare a datelor și prelucrarea lor se va executa într-un fir de execuție separat pentru a nu bloca interfața grafică. Clasa ce va implementa această funcționalitate se va numi DownloadWebpageTask și va extinde clasa abstractă AsyncTask. Aceasta va suprascrie două metode: doInBackground care va apela metoda de descărcare a adresei http trimise și onPostExecute care va apela metoda de prelucrare a datelor descărcate și primite ca parametru.

@Override  
**protected** String doInBackground(String... urls) {  
  
 **try** {  
 **return** downloadUrl(urls[0]);  
 } **catch** (IOException e) {  
 **return null**;  
 }  
}

**private** String downloadUrl(String myurl) **throws** IOException {  
 InputStream is = **null**;  
 **int** len = 10000;  
  
 **try** {  
 URL url = **new** URL(myurl);  
 HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
  
 is = **new** BufferedInputStream(conn.getInputStream());  
  
 String contentAsString = readStream(is);  
  
 **return** contentAsString;  
  
 } **finally** {  
 **if** (is != **null**) {  
 is.close();  
 }  
 }  
}

Metoda de prelucrare cuprinde extragerea datelor de interes din pagina HTML primită ca parametru sub forma unu șir lung de caractere și adăugarea timpilor de sosire pentru fiecare stație activă în parte. Tot aici se testează daca un mijloc de transport este pe ruta selectată între punctele de capăt, atunci se va afișa pe hartă poziția lui pe traseu.

## Interfața cu utilizatorul

Interfața cu utilizatorul reprezintă cea mai importantă parte a aplicației deoarece prin aceasta se redă informația dorită de utilizator. În acest sens, pentru acest proiect interacțiunea cu utilizatorul a ocupat un loc important în dezvoltare. Accentuând faptul că utilizatorul are nevoie de datele oferite de către aplicație și anume informații cu privire la locația personală, locația stațiilor și a timpilor de sosire în stație a transportului, aplicația afișează pe întreg ecranul harta Google, urmând în funcție de selecția utilizatorului să afișeze traseele liniilor selectate împreună cu locațiile pe hartă a stațiilor.

Pentru lista de selectare a liniilor s-a folosit un meniu tip „sertar” de navigație ce va fi afișat din partea stângă a ecranului. În acest fel activarea, schimbarea sensului sau dezactivatea unui traseu de pe hartă va fi foarte eficient din punct de vedere al timpului și al interacțiunii cu utilizatorul. Lista cu mijloacele de transport va fi afișată pe categorii, fiecare categorie având o anumită culoare astfel încât traseele de pe hartă să fie ușor diferențiate. După selectarea uneia dintre linii „sertar-ul” de navigație se va închide automat pentru a facilita viteza la accesul informațiilor dorite.

(SCREENSHOT MENIU 1. COLLAPSED 2. EXPANDED)

Stațiile corespunzătoare liniilor selectate se vor plasa pe hartă, fiind unite pe hartă cu o linie de culoarea mijlocului de transport ales. Culoarea fiecărei linii va avea o transparentă de 50% pentru ca atunci când două trasee se vor suprapune, suprapunerea să fie sesizată prin modificarea culorii pe acea porțiune de traseu. Linia se va generată cu ajutorul unui API, Google Maps Roads, ce va returna cel mai scurt traseu între două stații. Traseul va fi obținut în format XML și va conține o multitudine de puncte astfel încât linia afisată să fie desenată pe drum. Acest proces de primire a datelor cu privire la traseu se vor procesa într-un fir de execuție separat, după modelul implementat și la preluarea datelor de sosire a transportului în stație.

Pentru fiecare mijloc de transport selectat, vor fi afișate în parte, câte un buton cu funcția de actualizare a timpilor corespunzători fiecărei stații a traseului selectat. Tot pentru a nu obtura prea mult din hartă, fundalul butoanelor va fi alb cu o transparență de 20%.

(SCREENSHOT HARTA CU STATII )

## 

# Utilizarea aplicației

# Realizare. Punere in functie si Testarea Aplicatiei.

Aplicația a fost testată cu ajutorul unor persoane ce folosesc de obicei transportul în comun și sunt familiarizați cu sistemul de transport în comun și cu cel de informare al timpilor de călătorie. Prin această modalitate de testare a sistemului s-au putut corecta anumite erori de implementare.

O problemă sesizată a fost că atunci când traseele unor linii se suprapuneau pe o anumită porțiune, linia grafică ce unea stațiile între ele nu reflecta faptul că pe acea porțiune circulă două mijloace de transport ci era afișată doar o linie care a fost selectată prima dată. Rezolvarea acestei probleme s-a produs prin schimbarea transparenței liniilor de conectare a staiilor între ele, cu o transparență de 50%, astfel încăt daca două sau mai multe trasee ale mijloacelor de transport se vor suprapune atunci va fi vizibilă suprapunerea prin modificarea culorii traseului.

# Concluzii și direcții de continuare a dezvoltării

Comparând situția actuală a problemei, privind informarea publicului larg cu datele necesare folosirii mijloacelor de transport în comun, cu soluția implementată în această lucrare se pot observa o serie de îmbunătățiri ce sunt aduse actualului sistem prin folosirea acestei aplicații.

O primă îmbunătățire se poate observa la prima vedere prin faptul că acum se va cunoște și poziția reală, afișată pe hartă, a stației, informație ce nu se gasește în actuala aplicație Android.

O a doua îmbunătățire este în implementarea mecanismului de primire a timpilor de călătorie, aplicația obținând timpii pentru toate stațiile unui anumit traseu dintr-o singură accesare a unei locații internet, vechea aplicație accesând altă cale care returnează numai o stație pentru un anumit traseu la fiecare apelare fiind necesar un apel pentru fiecare stație a unui traseu fapt ce va întârzia timpul de răspuns cu privire la informațiile cerute de utilizator sau chair provocarea de blocări ale serverului din cauza accesării unui număr mare de utilizatori.

O altă îmbunătățire ar fi urmărirea în același timp a mai multor trasee pentru a folosi mijlocul de transport ce va ajunge cel mai rapid în punctul dorit de plecare și spre destinația dorită.

În urma unor discuții cu administratorul server-ului R.A.T.T pentru informarea folosirii datelor furnizate de ei în interiorul aplicației dezvoltate, mi-a lăsat la cunoștință faptul că există o pagină web cu toate stațiile în folosință, actualizată la cea mai mică modificare a unui traseu, pe care as putea să o accesez după depunerea unei cereri, în acest sens, la sediul regiei de transport. Beneficiul accesării acestei pagini pentru datele privind locația stațiilor si traseul unei linii, în locul fișierului CSV implementat în acest moment, ar fi că aplicația va furniza întotdeauna traseul actual al oricarei linii fără a mai fi nevoia de o actualizare manuală a rutelor modificate, acest lucru obținându-se automat.

O altă directie de dezvoltare ar fi calcularea automată a celei mai rapide modalitați de a ajunge dintr-o pozitie în alta și anume prin calcularea timpilor de sosire a fiecărui mijloc de transport corespunzător locației de plecare și a celei de destinație, apoi afișarea rutei si a combinațiilor de linii pentru a ajunge în cel mai rapid timp la destinație.

# Bibliografie

https://source.android.com/devices/

https://developer.android.com/studio/intro/index.html

<https://developers.google.com/maps/documentation/android-api/>

http://www.grokkingandroid.com/sqlite-in-android/

<https://www.sqlite.org/copyright.html>

<https://java.com/en/download/faq/whatis_java.xml>

1. https://java.com/en/about/ [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Java_(programming_language)> [↑](#footnote-ref-2)
3. https://en.wikipedia.org/wiki/Comma-separated\_values [↑](#footnote-ref-3)