**UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI**

# FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

**PublicTransportMonitor**

**propusă de**

**Adelin Pământ**

**Sesiunea:** ***Februarie, 2020***

**Coordonator ştiinţific**

***Prof. Colab. Florin Olariu***

**UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI**

**FACULTATEA DE INFORMATICĂ**

# PublicTransportMonitor

**Adelin Pământ**

**Sesiunea:** ***Februarie, 2020***

**Coordonator ştiinţific**

***Prof. Colab. Florin Olariu***

# DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul PublicTransportMonitor, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iaşi, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezenţei lucrări de licenţă.

Iaşi, data

Absolvent Prenume Nume

_____

(semnătura în original)

# CUPRINS

# INTRODUCERE

## 1. Motivaţie

Ideea aplicaţiei a pornit din dorinţa de a face mai plăcută şi de a uşura folosirea mijloacelor de transport public din Iaşi. Totodată aplicaţia ajută la optimizarea timpului necesar pentru a folosi mijloacele de transport public, aceasta minimizând timpul petrecut de aşteptare al unui autobuz sau al unui tramvai.

Transportul public este crucial într-un oraş, acesta aducând numeroase beneficii, de la accesibilitate din punct de vedere financiar, până la reducerea considerabilă a traficului şi implicit, a poluării aerului.

În timp ce transportul public aduce numeroase beneficii, serviciile oferite trebuie să se afle la un anumit nivel pentru a fi folosit de către publicul larg. Unul din cele mai importante criterii pe care serviciile de transport public trebuie să îl îndeplinească îl reprezintă accesul la informaţii legate de locaţia mijloacelor de transport public, astfel încât un utilizator să îşi poată organiza activităţile în funcţie de timpul necesar unui mijloc de transport să ajungă într-o staţie de interes pentru utilizator. Acest criteriu nu este îndeplinit de către serviciile de transport public din Iaşi, şi această aplicaţie a fost gândită şi construită cu scopul de a remedia acest neajuns.

## 2. Introducere în temă

Lucrarea realizată constă într-o aplicaţie din care un utilizator poate beneficia de numeroase funcţionalităţi a căror scop este de a îmbunătăţi experienţa folosirii mijloacelor de transport în comun.

Poate primi informaţii legate de autobuze sau tramvaie ce îi sunt utile pentru a se deplasa către orice destinaţie din Iaşi.

Are posibilitatea de a  vizualiza în timp real informaţii despre călătoria pe care o realizează prin intermediul unui mijloc de transport, utilizatorul ştiind în mod constant care este următoarea staţie în care va ajunge, câte staţii mai are de parcurs până la destinaţie şi când a ajuns la destinaţie.

O altă funcţionalitate, şi probabil cea mai importantă a acestei aplicaţii este aceea de a putea vizualiza informaţii legate de locaţia curentă a unui autobuz sau a unui tramvai de care un utilizator se poate folosi pentru a se deplasa către destinaţia dorită.

Aceasta din urmă fiind funcţionalitatea de bază a aplicaţiei, deoarece astfel este eliminat neajunsul ce constă în neştiinţa locaţiei oricărui autobuz sau tramvai, neajuns ce are ca şi consecinţă forţarea utilizatorului serviciilor de transport public să aloce îndeajuns timp de aşteptare în staţie astfel încât să poată ajunge la timp destinaţie. Beneficiul aceste funcţionalităţi este cu atât mai vizibil în situaţii în care timpul de care dispune o persoană este limitat şi optimizarea acestuia devine un factor extrem de important, precum un student ce trebuie să ajungă la facultate într-o zi de examen, a unei persoane ce trebuie să ajungă la un interviu şi aşa mai departe.

## 3. Structura lucrării

Lucrarea conţine trei mari capitole generale în care sunt detaliate mai multe informaţii despre lucrare. Per ansamblu, din acestea reies următoarele: impulsul ce m-a determinat să aleg această temă, ce îmbunătăţiri aduce soluţia propusă de mine în comparaţie cu soluţii deja existente, cât şi funcţionalităţile implementate în aplicaţie şi cum funcţionează acestea în spate.

În **primul capitol** este descrisă nu doar utilitatea, cât şi importanţa serviciilor de transport public într-un oraş modern, sunt detaliate beneficiile pe care acestea le aduc, dar şi neajunsurile acestora la momentul actual, neajunsuri care îi determină de multe ori pe oameni să aleagă alte variante în favoarea mijloacelor de transport în comun când vine vorba de deplasarea dintr-un loc în altul în cadrul oraşului.

În **al doilea capitol** sunt prezentate câteva aplicaţii deja existente ce au abordat problema descrisă în Capitolul 1 şi cu ce rezolvări au venit. Sunt detaliate funcţionalităţile implementate în aceste soluţii, îmbunătăţirile pe care acestea le aduc în ceea ce priveşte transportul public, cât şi funcţionalităţi pe care eu le consider esenţiale şi lipsesc din aceste soluţii.

În **al treilea capitol** este descrisă abordarea mea atât de a rezolva problema descrisă în primul capitol cât şi de a umple golurile din aplicaţiile descrise în al doilea capitol. Este descrisă ideea generală a proiectului, arhitectura acestuia, alături de funcţionalităţile principale şi cum sunt implementate acestea.

În final, se pot regăsi concluziile trase de mine în legătură cu această soluţie şi implementarea acesteia.

# CONTRIBUŢII

Numeroase modalităţi au fost încercate de-a lungul timpului pentru rezolvarea problemelor transportului public. Majoritatea oferă utilizatorilor detalii privind mijloacele de transport pe care le pot folosi pentru a ajunge la destinaţia dorită, şi totodată oferă o aproximare a momentului în care acele mijloace de transport vor ajunge într-un loc de interes pentru utilizator. Şi în timp ce această aproximare poate fi o adiţie foarte utilă, din păcate este de cele mai multe ori departe de a fi precisă, rezultând într-o funcţionalitate care în cele din urmă ajunge să încurce mai mult decât sa ajute şi în final să nu mai fie folosită deloc.

Soluţia propusă de PublicTransportMonitor este de a nu se axa pe acest tip de informaţie, ci pe informaţii concrete astfel încât utilizatorul să poată deduce când va ajunge un mijloc de transport într-un loc de interes pentru acesta. Utilizatorul nu va putea deduce când va ajunge mijlocul de transport cu o precizie la nivel de minut, însă va şti staţiile deja parcurse de acesta, şi astfel va putea deduce cu uşurinţă când şi unde va ajunge un mijloc de transport.

# CAPITOLUL 1 - DESCRIEREA PROBLEMEI

Deplasarea folosind mijloacele de transport în comun aduce numeroase beneficii, atât locuitorilor oraşelor cât şi mediului înconjurător. Pentru noi, în calitate de utilizatori beneficiile sunt numeroase. Din punct de vedere al cheltuielilor, folosirea mijloacelor de transport în comun este de departe cea mai accesibilă. Un beneficiu pentru mediul înconjurător, şi implicit şi pentru noi, este faptul că folosirea cât mai mult a mijloacelor de transport în comun va rezulta în reducerea drastică a traficului în oraşe; astfel, oamenii nu vor mai fi blocaţi cu orele pe drum, nivelul de poluare se va reduce considerabil, fapt ce în final tot pentru utilizatori reprezintă un mare avantaj.

Enumerarea de avantaje poate continua, însă ce consider mai important este ce anume lipseşte cand vine vorba de servicii de transport în comun dacă în timp ce aduc numeroase avantaje de luat în considerare, majoritatea oamenilor sunt reticenţi în a le folosi.

Problema principală când vine vorba de deplasarea folosind mijloacele de transport în comun este faptul că publicul nu are acces la locaţia acestora şi astfel folosirea lor presupune luarea în calcul a timpului de aşteptare în staţii, timp ce poate fi insignifiant, cum ar fi 2 minute, dar poate deveni şi o mare problemă atunci când ajunge şi la 30 de minute. Şi mai problematică decât aşteptarea timp de 30 de minute pentru un mijloc de transport este variaţia timpului de aşteptare, pentru că din cauza acestei probleme oamenii nu pot să îşi planifice într-un mod cât mai eficient posibil timpul şi sunt nevoiţi să-şi ia o marjă de eroare uneori chiar mare în funcţie de cât de important este pentru aceştia să ajungă la timp la destinaţie.

# CAPITOLUL 2 - ABORDĂRI ANTERIOARE

## 2.1 Moovit

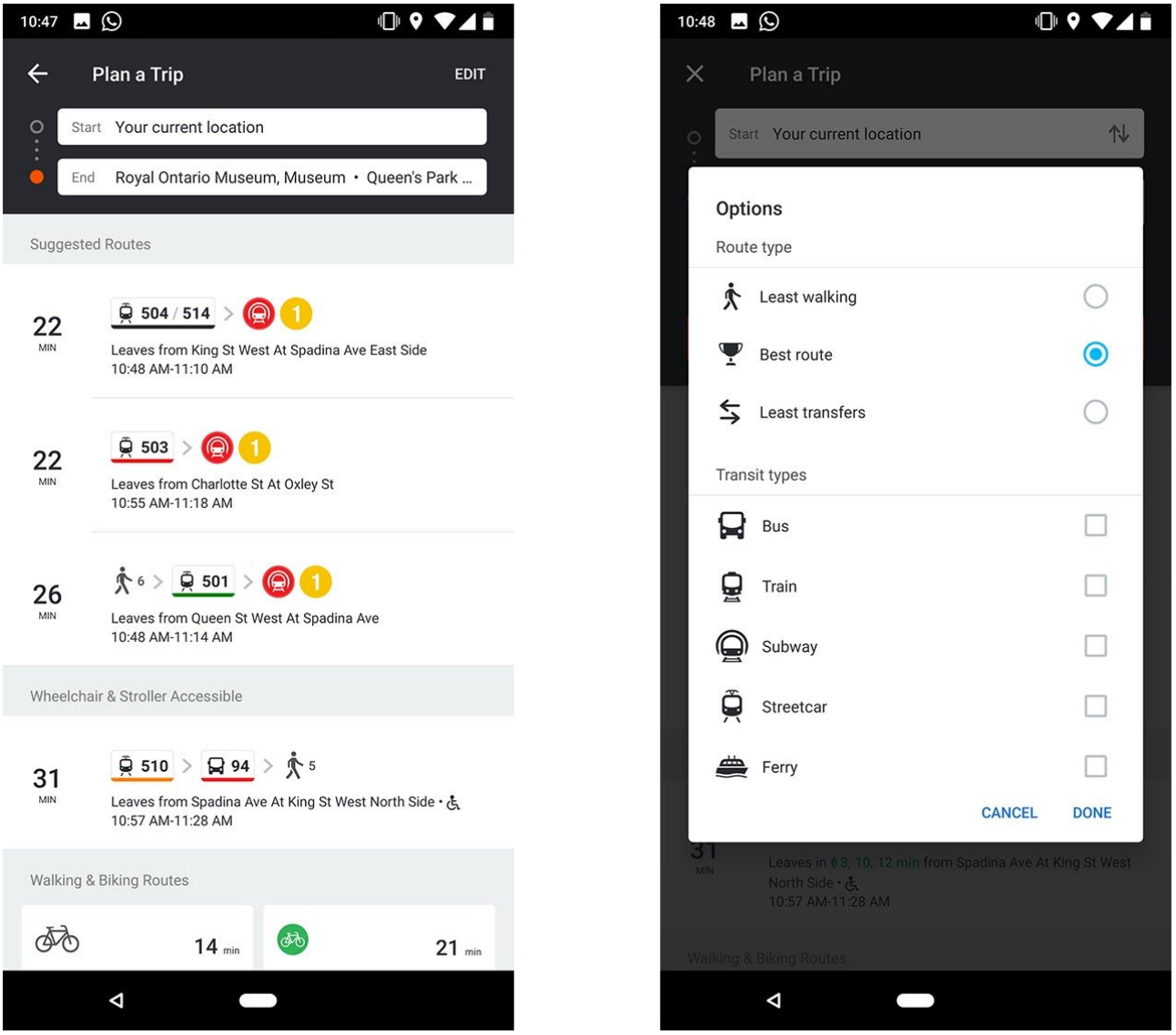


Figura 1: Aplicația Moovit [1]

Moovit este una din cele mai populare soluţii de îmbunătăţire a serviciilor de transport public. Această aplicaţie oferă utilizatorilor informaţii despre mijloacelor de transport pe care le pot folosi pentru a se deplasa. Aceasta are ca şi avantaje faptul că oferă informaţii legate de trenuri, metrouri, etc. De asemenea oferă informaţii despre timpul aproximativ în care va ajunge un mijloc de transport într-o anumită staţie.

Problema cu această aproximare o reprezintă posibilitatea apariţiei de accidente, trafic, şi multe altele care dau peste cap această aproximare, şi astfel utilizatorul nu are la dispoziţie informaţii sigure legate de mijloacele de transport.

## 2.2 Google Maps

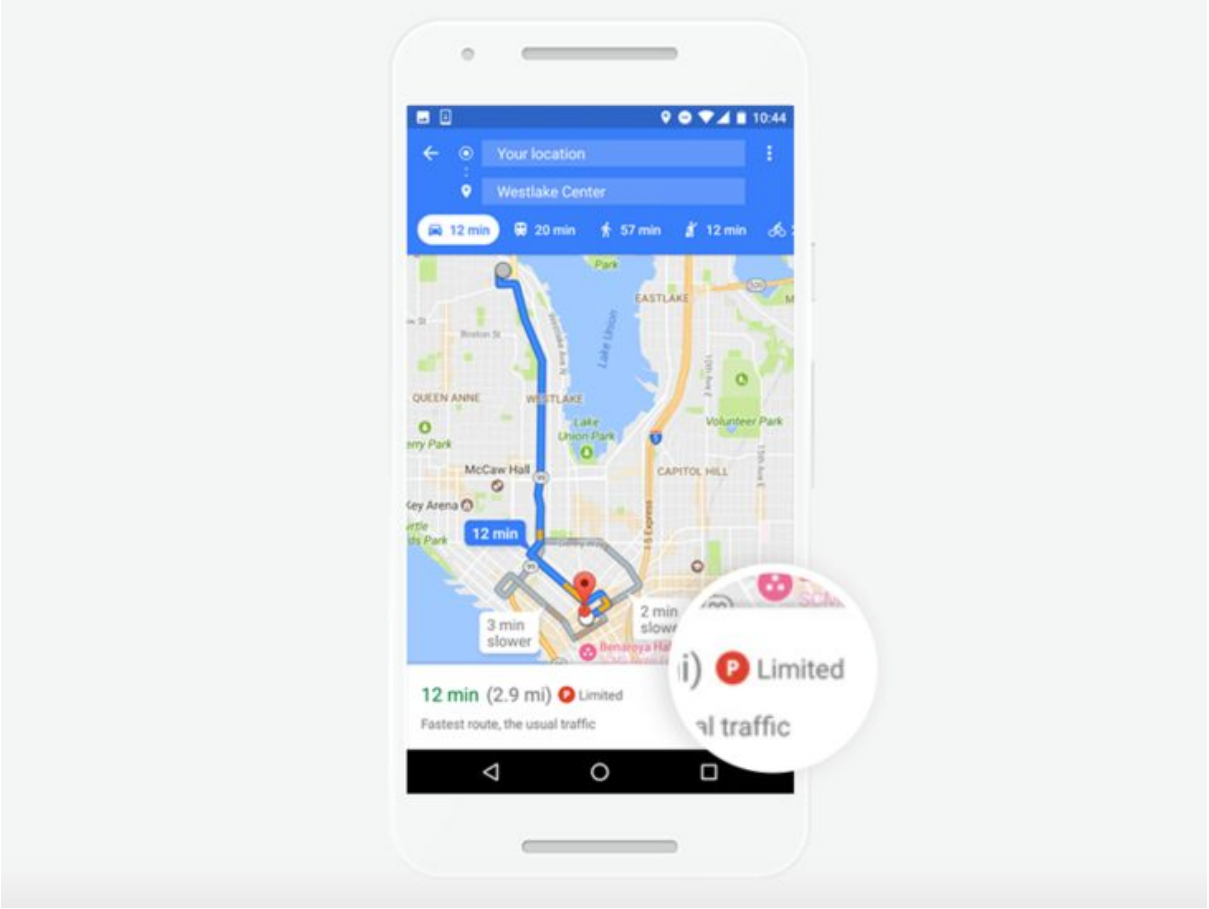


Figura 2: Aplicaţia Google Maps [2]

Google Maps este probabil cea mai folosită aplicaţie pentru deplasarea folosind mijloacele de transport în comun. Beneficiile pe care această aplicaţie le aduce sunt: posibilitatea de a vedea ce mijloace de transport putem folosi pentru a ne deplasa dintr-un loc în altul, inclusiv pe care le putem împreună pentru a merge pe cea mai eficientă rută.

Probabil factorul care face această aplicaţie să fie atât de folosită este că pe lângă faptul că livrează la o calitate de nivel înalt serviciile pe care le oferă, vine preinstalată pe majoritatea telefoanelor, tabletelor şi este disponibilă şi în browser.

Problema însă cu această aplicaţie este aceeaşi cu cea de la Moovit; şi anume că deşi oferă şi aproximări legate de când va ajunge un mijloc de transport în staţie, tot nu oferă informaţii ce sunt 100% valide cu privire la acest aspect.

## 3.3 HEREITIS

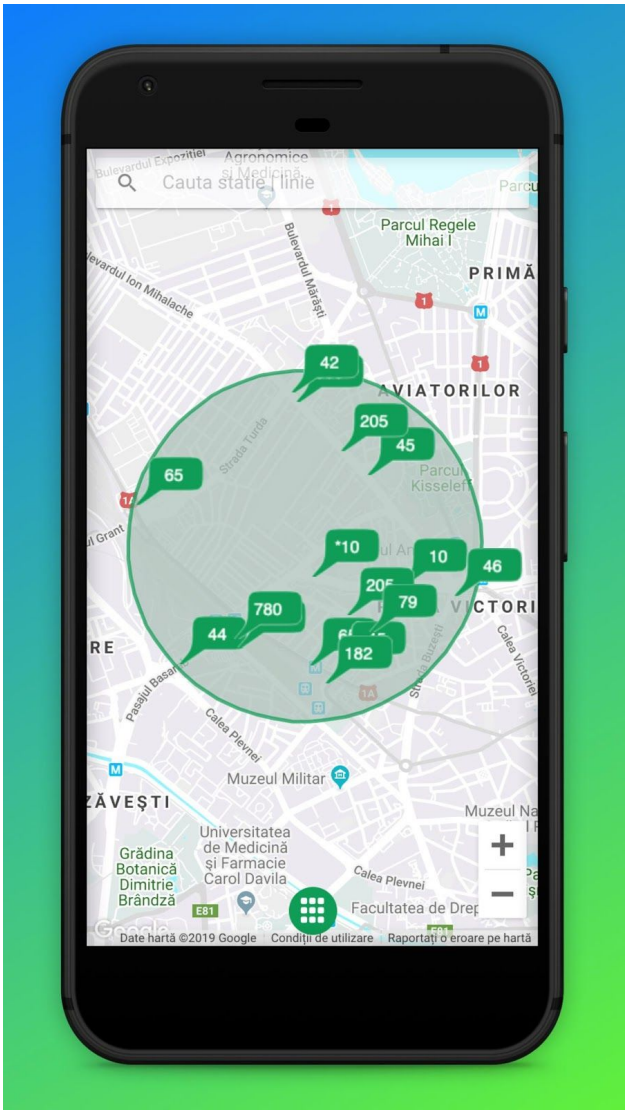


Figura 3: Aplicaţia HEREITIS [3]

HEREITIS este una din cele mai noi soluţii pentru rezolvarea neajunsurilor serviciilor de transport public. Aceasta a pornit cu o idee similară cu PublicTransportMonitor, având la bază principiul de a lăsa la latitudinea utilizatorilor să înregistreze un mijloc de transport. Ulterior, aplicaţia a trecut de la această funcţionalitate la folosirea unor date expuse de CTP ce ţin de locaţia curentă a mijloacelor de transport.

Aplicaţia este una reuşită, însă prezintă unele aspecte ce pot fi îmbunătăţite, cum ar fi:

- Interfaţa: nu mă refer aici la aspectul acesteia ci la uşurinţa de a o folosi, de a găsi ceea ce caut
- Filtrare: HEREITIS afişează la un moment dat toate mijloacele de transport pe o rază de 10 km., ceea ce face îngreunează utilizatorul din a urmări doar informaţiile de interes pentru acesta

Este de discutat dacă folosirea datelor expuse de CTP este modalitatea mai bună de a expune date în aplicaţie deoarece acele date vin cu anumite limitări, cum ar fi: posibilitatea de a actualiza datele este limitată la posibilitatea de a expune date de către CTP, iar acurateţea datelor nu poate fi constantă, acest fapt fiind o consecinţă a felului în care sunt expuse datele legate de mijloacele de transport.

PublicTransportMonitor acoperă aceste neajunsuri prin a oferi utilizatorilor posibilitatea de a filtra informaţiile afişate, iar datele legate de mijloacele de transport sunt calculate şi afişate în funcţie locaţia utilizatorului şi datele introduse de acesta.

# CAPITOLUL 3 - DESCRIEREA SOLUŢIEI

## 3.1 Descriere proiect

Proiectul realizat este o aplicaţie în care orice utilizator are posibilitatea fie de a căuta mijloace de transport în comun a căror locaţie este vizibilă pentru utilizator, fie să contribuie la aplicaţie prin a înregistra un mijloc de transport în comun pentru a îi face vizibilă locaţia pentru ceilalţi utilizatori ai acesteia.

Funcţionarea aplicaţiei se bazează pe implicarea comunităţii. Din acest motiv a fost concepută într-un mod cât mai simplist din punct de vedere al interfeţei. În acelaşi timp au fost luate în calcul uşurinţa de a fi folosită şi oferirea de informaţii cât mai concise dar utile în acelaşi timp, utilizatorul având posibilitatea de a accesa informaţii mai detaliate dacă doreşte. Interfaţa a fost gândită să fie cât mai simplă deoarece cum aplicaţia se bazează pe utilizatorii acesteia utilizarea ei trebuie să fie cât mai uşoară şi intuitivă.

În calitate de utilizator al aplicaţiei, două roluri sunt posibile.

Unul dintre acestea este cel de utilizator care înregistrează un mijloc de transport în comun. Tot ce trebuie acesta să facă este să introducă în aplicaţie staţia din care porneşte, staţia în care vrea să ajungă şi numele mijlocului de transport cu care doreşte să meargă. Pentru a facilita utilizarea aplicaţiei, o listă cu numele fiecărei staţii este afişată când utilizatorul începe să scrie numele unei staţii, lista respectivă conţinând din ce în ce mai puţine staţii pe parcurs ce utilizatorul scrie, şi orice element din listă poate fi selectat pentru a fi scris automat. Apoi, cu acordul acestuia, locaţia utilizatorului va fi persistată şi implicit, locaţia mijlocului de transport în care se află. După ce a înregistrat călătoria, şi implicit, locaţia mijlocului de transport, utilizatorul va putea urmări progresul călătoriei acestuia până va ajunge la destinaţie.

Celălalt rol posibil este acela de utilizator care foloseşte informaţiile expuse de către celălalt. Acest utilizator îşi va introduce staţia de plecare şi staţia în care doreşte să ajungă. Rezultatul căutării fiind toate mijloacele de transport a căror locaţie este expusă print alţi utilizatori şi a căror traseu se potriveşte cu cel al utilizatorului. De asemenea staţiile pe care utilizatorul trebuie să le introducă pot fi completate în mod automat urmărind aceeaşi procedură cu cea descrisă ulterior pentru utilizatorul care expune date despre un mijloc de transport. Dacă doreşte, utilizatorul are de asemenea posibilitatea de a deschide o fereastră cu mai multe detalii legate de un anumite mijloc de transport de care este interesat în mod special.

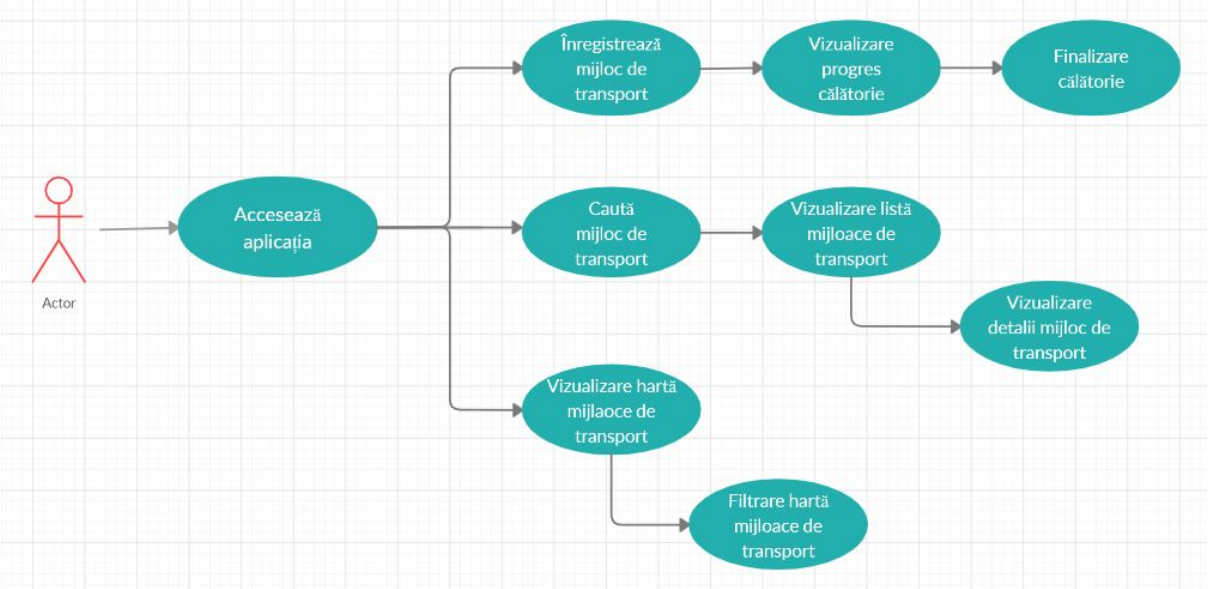


Figura 4: Scenariile de utilizare ale aplicaţiei

## 3.2 Arhitectura proiectului

În diagrama de mai jos este prezentat felul în carea fost proiectată aplicaţia.

Elementele principale ale aplicaţiei sunt:

- **Utilizatorul:** ce poate fi oricine are acces la internet şi este de acord ca aplicaţia să îi acceseze locaţia

- **Aplicaţia web:** expune funcţionalităţile de a înregistra sau căuta mijloace de transport
- **Serverul:** orchestrează datele primite de la aplicaţia web pentru a calcula locaţiile tuturor mijloacelor de transport curente şi de a le expune
- **HTML Geolocation API:** API-ul folosit la nivel de client pentru a cere date legate de locaţia curentă a unui utilizator

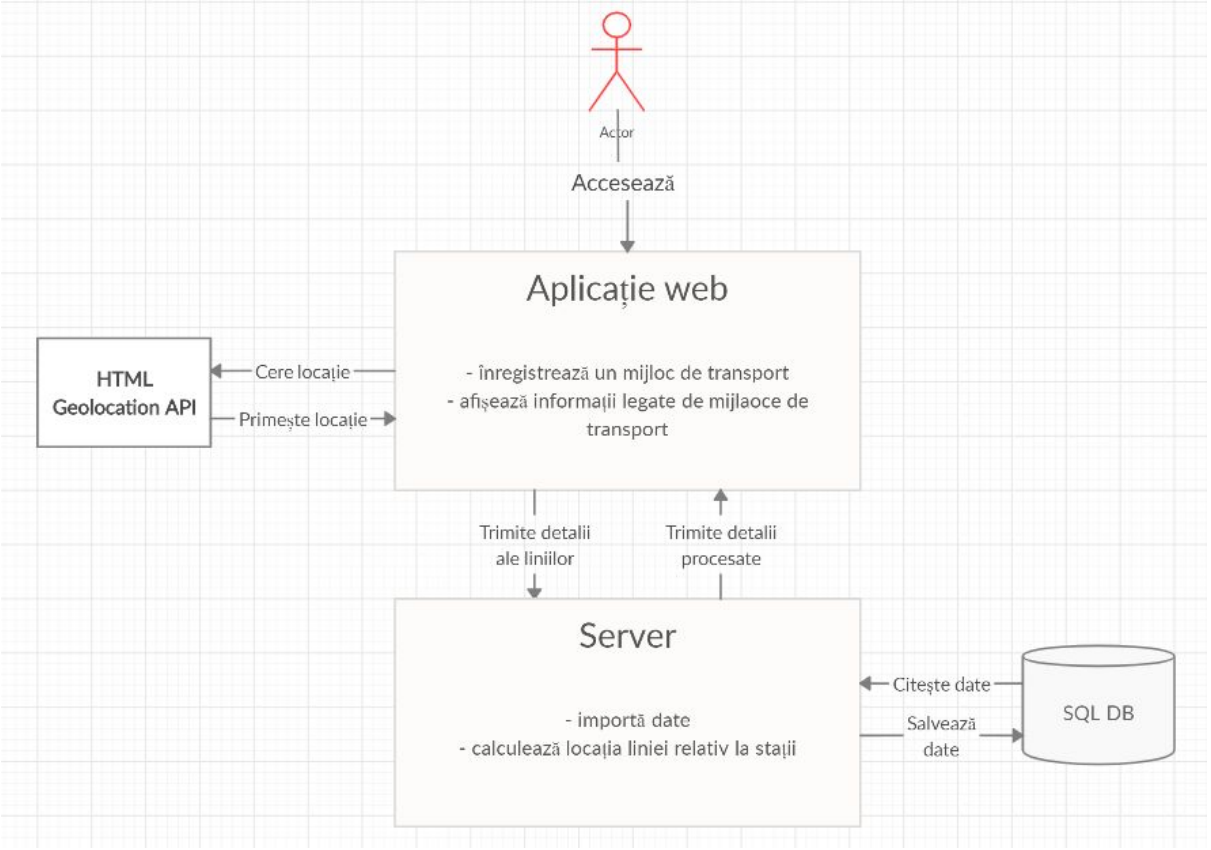


Figura 5: Diagramă contextuală

### 3.2.1 Arhitectură frontend

Pentru client am ales să construiesc o aplicaţie web folosind Angular, deoarece astfel elimin problema compatibilităţii pe diferite platforme. Aplicaţia la nivel de client este construită în concordanţă cu Angular Styleguide. Aceasta este împărţită în componente, în cadrul cărora au fost respectate standardele pentru construirea unei aplicaţii Angular.

Logica aplicației este încapsulată în servicii, fiecare serviciu respectând principiul Single Responsibility. Aceste servicii sunt apoi injectate în componente folosind mecanismul de Dependency Injection încorporat în Angular.

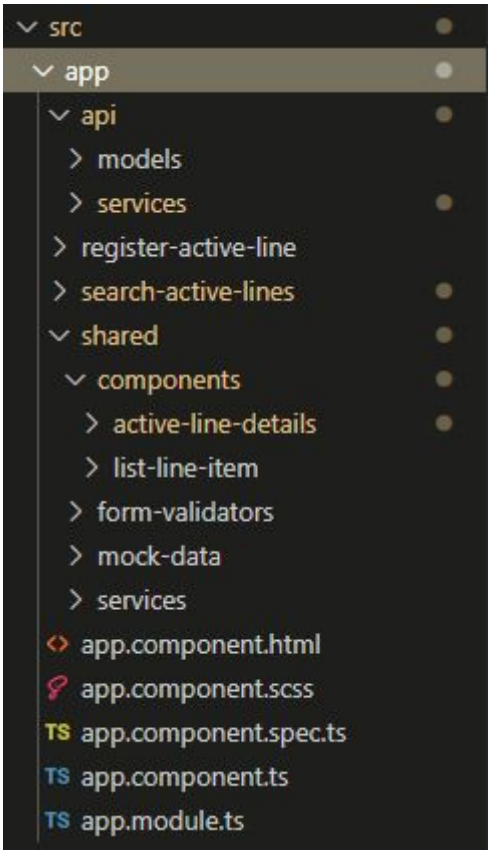


Figura 6: Soluția de frontend

Serviciile sunt de asemenea împărțite în 2 categorii: servicii ce țin de interacționarea cu API-ul construit; tot ce înseamnă citire și trimitere de date către server, și servicii ce țin de logica implementată în frontend, cum ar fi serviciul pentru orchestrarea funcționalităților legate de locația unui utilizator.

De asemenea au fost respectate principii și practici folosite pentru a scrie cod Typescript, cum ar fi: folosirea de tipuri primitive, definirea concretă a tipurilor folosite fie tipuri de parametri, fie tipul de returnare așteptat al unei metode, folosirea de tipuri Union, etc.

### 3.2.2 Arhitectură backend

Soluția de backend este împărțită în 2 mari componente:

- ImportModule: ce ține de partea de importare a datelor de pe site-ul CTP
- MainModule: ce ține de partea de logică a aplicației în sine

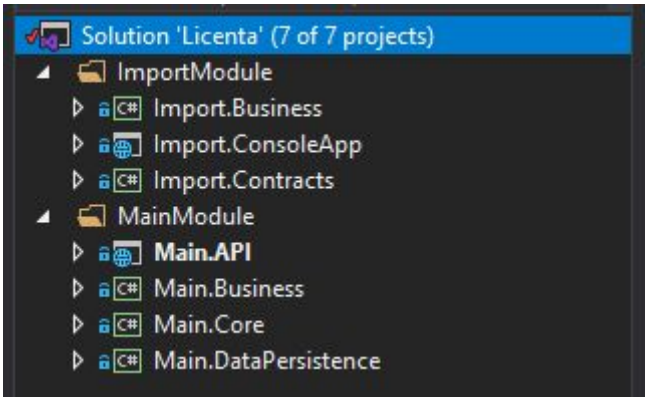


Figura 7: Soluția de backend

În componenta ImportModule se află toată logica pentru aducerea datelor de pe site-ul CTP, maparea acestora la obiecte definite în proiectul Import.Contracts pentru a putea fi persistate în baza de date, şi introducerea acestor date în baza de date. Această componentă poate fi extinsă pe viitor pentru a importa date din diferite endpoint-uri, singurele lucruri necesare fiind respectivele endpoint-uri şi contractele care trebuie respectate.

În componenta MainModule se află logica efectivă a aplicației. Modulul este structurat astfel: Main.API este API-ul expus cu funcționalitățile necesare, acesta folosește servicii implementate în Main.Business, proiect care la rândul lui respectă contracte pentru repository-urile definite în Main.Core. Serviciile folosite în Main.Core sunt wrappere pentru repository-urile implementate pentru persistarea datelor în baza de date, acestea executând diferite metode ce țin de procesarea datelor înainte de a le scrie în baza de date sau de a le trimite către client.

Pentru operațiunile de scriere în baza de date sau citire din aceasta, nu au fost folosite entitățile definite pentru fiecare obiect, ci DTO-uri specifice fiecărei acțiuni ale unei entități. Astfel, este separată logica pentru persistarea datelor de cea pentru procesarea acestora, şi viitoare schimbări necesare pentru operații de scriere/citire din baza de date vor presupune schimbări la nivelul DTO-urilor, dar nu şi la nivelul entităților acestora, maparea dintre entități şi DTO-uri fiind procesată folosind AutoMapper.

## 3.3 Implementarea proiectului

### 3.3.1 Înregistrarea unui mijloc de transport

Orice utilizator al aplicaţiei poate înregistra un mijloc de transport. Aceasta este funcţionalitatea pe care se bazează această aplicaţie. Pentru a înregistra un mijloc de transport, utilizatorul trebuie să navigheze către pagina **Register Line** şi să introducă în aceasta staţia din care porneşte, staţia în care doreşte să ajungă, şi numele mijlocului de transport în care s-a urcat/se urcă. Pentru a uşura sarcinile utilizatorului, atunci când începe să scrie numele unei staţii, o listă cu toate opţiunile acestuia vor fi afişate. Această listă se scurtează pe parcurs ce utilizatorul completează câmpul respectiv, şi orice staţie din listă poate fi selectată şi ca rezultat numele acesteia este completat în mod automat în câmpul respectiv.


Figura 8: Formular de înregistrare al unei linii

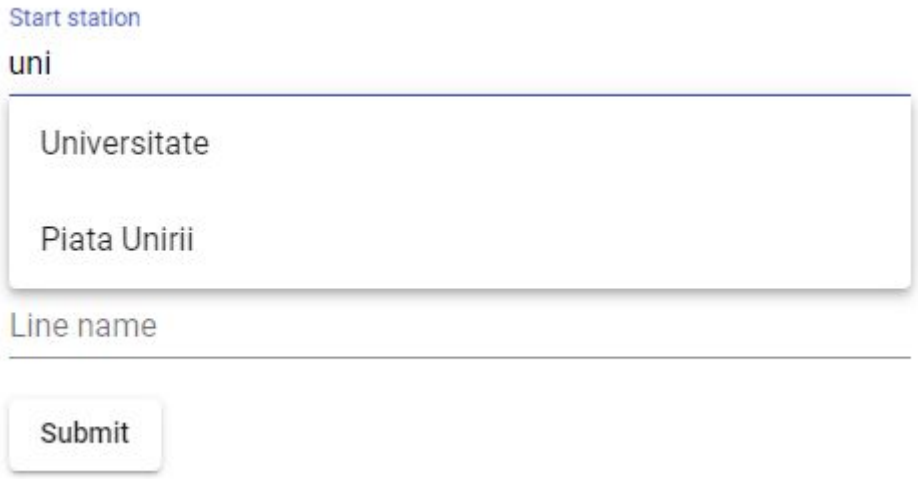


Figura 9: Funcţionalitatea de auto completare

În frontend, aceste informaţii sunt centralizate şi procesate astfel:

```
private handleLineRegister(): void {
  this._locationService.startTracking((position: Position) => {
    let activeLineForCreation = this.buildActiveLine(position);
    let activeLineForPatch = this.buildActiveLineForPatch(position);

    this._subscription = this._activeLineService.getActiveLineObservableByUri(
      this._registeredLineUri,
      activeLineForCreation,
      activeLineForPatch
    )
    .subscribe((response: HttpResponse<ActiveLineForCreation> | ActiveLine) => {
      // null check because PATCH returns NoContent() and thus, null ?
      if (response !== null &&
          'headers' in response) {
        this._registeredLineUri = response.headers.get('location');
        this._subscription.unsubscribe();
      }
    });
  });
}
```

Figura 10: Implementarea pentru înregistrare şi urmărire a unei linii

1. În primă instanţă, este apelată metoda **startTracking** ce aparţine de **LocationService**. În cadrul acesteia este apelată metoda **watchPosition** din Geolocation API. Funcţia apelată în cazul în care locaţia utilizatorului este identificată cu succes este cea definită în **handleLineRegister**, după cum putem observa în figura de mai sus.

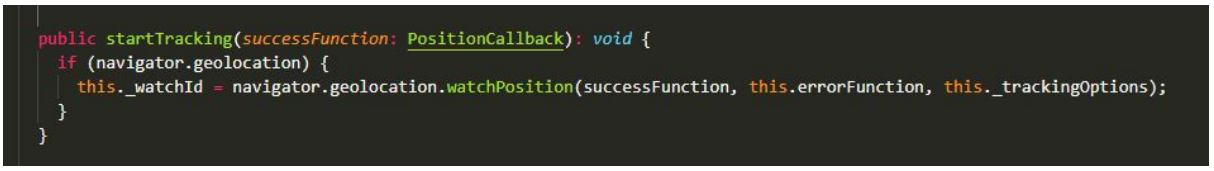


Figura 11: Implementarea metodei de urmărire a API-ului

2. Pasul următor este reprezentat de crearea a două obiecte diferite, unul pentru a crea o nouă linie, celălalt pentru a modifica linia deja creată. Ambele obiecte sunt create în cadrul acestei metode deoarece pe tot parcursul călătoriei utilizatorului locația acestuia este urmărită prin metoda **startTracking**, în primă instanță linia va fi creată, URI-ul acestei linii va fi salvat local și la acel URI se vor face modificările cu obiectul creat special pentru această operație
3. Logica ce ține de decizia de a adăuga o nouă linie sau de a modifica una existentă este încapsulată în metoda **getActiveLineObservableByUri** din **ActiveLineService**.

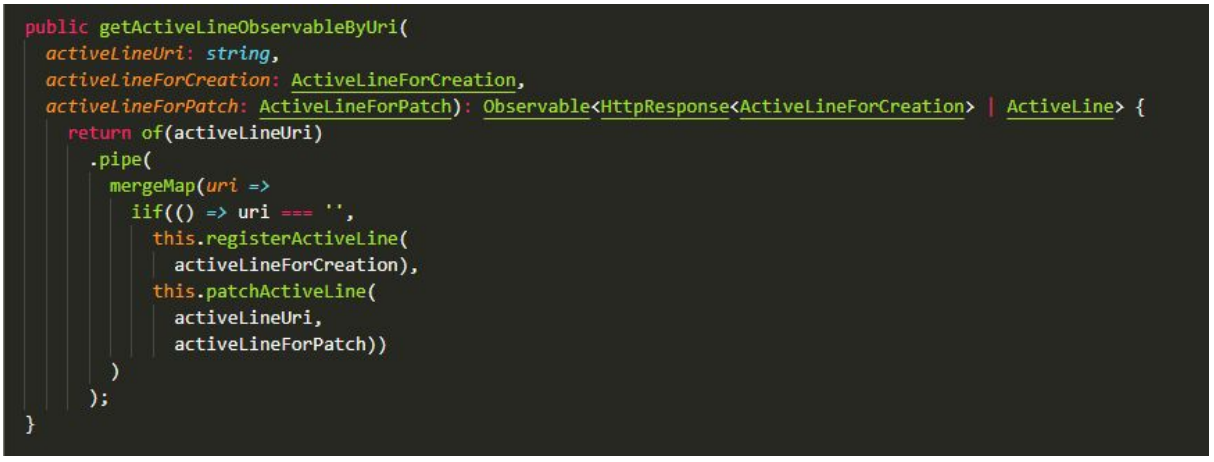


Figura 12: Implementare alegere dinamică a acțiunii pe linie

Aceasta se folosește de operatorul **RxJS iif** pentru a apela ori metoda de creare definită în serviciu, ori pe cea de modificare.

În backend, informațiile primite de la frontend sunt procesate astfel:

1. În cazul creării unei noi linii, informațiile primite de la frontend sunt folosite pentru a identifica toate detaliile liniei introduse de

către utilizator. Totodată sunt adăugate detalii calculate specifice pentru linia adăugată

```
2 references | Adelin Pamint, 4 days ago | 1 author, 4 changes
public Guid CreateActiveLine(ActiveLineForInsertDto activeLineForInsertDto)
{
    var activeLine = _mapper.Map<ActiveLine>(activeLineForInsertDto);
    activeLine.NextStation = _activeStationService.GetNextStationOf(activeLine).Name;
    activeLine.NextStationStatus = NextStationStatus.JustLeftStation;

    _activeLineRepository.AddActiveLine(activeLine);
    _activeLineRepository.SaveChanges();

    return activeLine.Id;
}
```

Figura 13: Crearea liniei pe server

În cazul unei modificări a unei linii existente, distanţa parcursă de aceasta se calculează folosind formula Haversine. În funcţie de rezultatul calculelor, detaliile liniei sunt modificate.

```
2 references | Adelin Pamint, 4 days ago | 1 author, 1 change
private decimal HaversineDistanceInKm(decimal currentLatitude, decimal currentLongitude, decimal latitude, decimal longitude)
{
    double radius = 6371;
    var lat :double = (double)(latitude - currentLatitude) * (Math.PI * 180);
    var lng :double = (double)(longitude - currentLongitude) * (Math.PI * 180);
    var h1 :double = Math.Sin(lat / 2) * Math.Sin(lat / 2) +
             Math.Cos((double)currentLatitude * (Math.PI * 180)) *
             Math.Cos((double)latitude * (Math.PI * 180)) *
             Math.Sin(lng / 2) * Math.Sin(lng / 2);
    var h2 :double = 2 * Math.Asin(Math.Min(1, Math.Sqrt(h1)));

    return (decimal)(radius * h2);
}
```

Figura 14: Metoda pentru calcularea distanţei între puncte

### 3.3.2 Căutarea unui mijloc de transport

Orice utilizator poate căuta mijloace de transport de interes pentru acesta. Pentru a putea efectua o căutare, utilizatorul trebuie să introducă numele staţiei din care doreşte să pornească şi numele staţiei în care doreşte să ajungă. Pentru a facilita sarcinile pe care acesta trebuie să le îndeplinească, atunci când utilizatorul începe să scrie numele unei staţii, o listă cu toate staţiile disponibile îi este afişată.

Această listă este filtrată în mod constant în funcţie de caracterele introduse de către utilizator, şi orice staţie din listă poate fi selectată pentru a completa în mod automat un câmp.

În funcţie de rezultatele căutării, o listă de informaţii legate de liniile ale căror locaţie este expusă de către alţi utilizatori este afişată.


Figura 15: Formular pentru căutarea unei staţii

În frontend, informaţiile introduse de către utilizator sunt centralizate şi procesate astfel:

```
protected submit() {
  this._activeLineService.getActiveLines(
    this.stationsForm.get('startStation').value,
    this.stationsForm.get('destinationStation').value
  ).subscribe(
    (data: ActiveLine[]) => this.activeLines = data
  );
}
```

Figura 16: Implementarea în frontend a căutării de linii

1. În primă instanţă este apelată metoda **getActiveLines** din serviciul **ActiveLineService** şi primeşte drept parametri staţia de pornire introdusă de utilizator, şi pe cea în care acesta doreşte să ajungă
2. Pasul al doilea constă în salvarea într-o variabilă locală a datelor recepţionate de la server, date ce vor fi ulterior folosite pentru a afişa utilizatorului informaţii legate de căutarea acestuia

Informaţiile afişate utilizatorului reprezintă linii ale căror locaţie este cunoscută în aplicaţie. Acestea sunt afişate în modul următor:


Figura 17: Detaliile afişate utilizatorului în urma căutării

În backend, informaţiile primite de la frontend sunt procesate astfel:

1. Sunt căutate în baza de date toate liniile ce corespund cu datele introduse de utilizator. Liniile rezultate sunt mapate la **ActiveLineForDisplayDto**, astfel, doar informaţiile necesare afişării în fronted sunt expuse.

```
2 references | Adelin Pamint, 14 days ago | 1 author, 2 changes
public IEnumerable<ActiveLine> GetActiveLines(ActiveLineResourceParameters activeLineResourceParameters)
{
    if (string.IsNullOrEmpty(activeLineResourceParameters.StartStation) ||
        string.IsNullOrEmpty(activeLineResourceParameters.DestinationStation))
    {
        return GetActiveLines();
    }

    return GetActiveLines()
        .Where(activeLine => HasStationsInGivenOrder(activeLine, activeLineResourceParameters)) // IEnumerable<ActiveLine>
        .ToList(); // List<ActiveLine>
}
```

Figura 18: Căutarea liniilor în baza de date

### 3.3.3 Vizualizare detalii linie

Componenta de detalii a unei linii presupune, în acest caz o secţiune în care sunt afişate următoarele: numele liniei, traseul pe care aceasta îl urmează, astfel încât utilizatorul să poată deduce direcţia în care merge linia respectivă, staţiile acesteia, fiecare staţie fiind marcată în 4 modalităţi posibile:

- **Passed:** semnifică faptul că linia a cărei detalii sunt urmărite a trecut deja prin staţia respectivă
- **Current:** semnifică faptul că linia este staţionată în respectiva staţie
- **Just passed**: acest marcaj se referă la faptul că linia a plecat de curând din acea staţie. Distanţa parcursă este luată în calcul pentru definirea acestui marcaj

- **Within line range**: acest marcaj este predestinat situaţiilor în care linia este în apropierea următoarei staţii

Fiecare staţie are, de asemenea, ora la care i-a fost atribuită un anumit marcaj.

Detaliile unei linii pot fi vizualizate atât de un utilizator care a înregistrat o anumită linie, cât şi de un utilizator care caută anumite linii în aplicaţie. Scopul acestei funcţionalităţi are, însă, diferite beneficii în funcţie de tipul de utilizator care o foloseşte.

Un utilizator care a înregistrat o linie are ca beneficii rezultate din această funcţionalitate faptul că poate vedea un progres live al călătoriei acestuia. Poate vedea atât staţiile pe care le-a parcurs, cât şi pe cele pe cele care i-au mai rămas. Astfel, utilizatorul poate face o aproximare a timpului pe care îl mai are de parcurs.

Un utilizator care caută linii, beneficiază de această funcţionalitate prin prisma faptului că poate vizualiza detaliile unei singure linii care îi este folositoare, eliminând, astfel, efortul suplimentar pe care trebuie să îl depună pentru a urmări o singură linie într-o colecţie cu număr variabil de linii. De asemenea, în cazul în care utilizatorul nu îşi poate da seama de locaţia liniei doar din staţia în care se află sau care este următoarea staţie în care urmează să ajungă linia, utilizatorul se poate folosi de modul detaliat de vizualizare a liniei pentru a vedea câte staţii sunt întrea ce din care acesta doreşte să plece, şi cea în care sau spre care se îndreaptă linia.

### 3.3.4 Vizualizarea liniilor pe hartă

Această componentă presupune vizualizarea pe o hartă a tuturor liniilor a căror locaţie este expusă prin intermediul locaţiei utilizatorilor. Acest mod de vizualizare vine în întâmpinarea utilizatorilor ce sunt obişnuiţi cu o vizualizare mai amplă a detaliilor expuse în aplicaţie.

Detaliile puse la dispoziţia utilizatorilor în acest mod de vizualizare sunt următoarele:

- Harta cu împrejurimile, astfel utilizatorul poate vedea direct unde se află o anumită linie
- Liniile active şi poziţia lor exactă

Liniile vizibile pe hartă pot fi filtrate după nume, astfel în cazul în care harta devine prea aglomerată pentru a putea fi folosită eficient, utilizatorul poate reduce informaţiile vizibile doar la cele de interes pentru acesta.

Informaţiile afişate pentru fiecare linie pe hartă sunt următoarele: numele acesteia şi traseul pe care îl parcurge, astfel utilizatorul poate vedea dacă linia respectivă merge pe un traseu de interes pentru acesta sau nu.

### 3.3.5 Stocarea datelor folosind SQL şi Entity Framework Core

Pentru stocarea datelor am ales să folosesc ORM-ul Entity Framework Core împreună cu o bază de date SQL. Am ales să folosesc EF Core deoarece acest framework vine cu toate funcţionalităţile necesare pentru a manageria persistenţa datelor într-o bază de date.

Logica folosită împărţirea tabelelor şi a entităţilor este asemănătoare cu cea aplicată la împărţirea logicii aplicaţiei astfel încât au rezultat 2 categorii:

- tabelele şi entităţile folosite pentru a importa datele de pe site-ul CTP
- tabelele şi entităţile folosite pentru a ţine evidenţa liniilor a căror locaţie este expusă de către utilizatori

Tabelele şi entităţile din cele 2 categorii nu sunt conectate în vreun fel, acestea fiind complet separate.

Entităţile folosite pentru logica aplicaţiei sunt bazate pe cele pentru a importa date, însă au diferite modificări pentru a se plia pe nevoile aplicaţiei.

```
1 reference | Adelin Pamint, 49 days ago | 1 author, 1 change
public DbSet<Line> Lines { get; set; }

0 references | Adelin Pamint, 49 days ago | 1 author, 1 change
public DbSet<Route> Routes { get; set; }

1 reference | Adelin Pamint, 49 days ago | 1 author, 1 change
public DbSet<Station> Stations { get; set; }

2 references | Adelin Pamint, 49 days ago | 1 author, 1 change
public DbSet<ActiveLine> ActiveLines { get; set; }

0 references | Adelin Pamint, 49 days ago | 1 author, 1 change
public DbSet<ActiveStation> ActiveStations { get; set; }
```

Figura 19: Entităţile folosite în aplicaţie

Entităţile create pentru a importa datele sunt create pe baza datelor expuse de către site-ul CTP, modelul acestora fiind asemănător cu JSON-urile expuse pe site, această decizie a fost luată pentru a minimiza nivelul de confuzie în cazul comparării datelor expuse pe site cu cele importate în aplicaţie. Aceste entităţi sunt împărţite astfel: Line, Station şi respectiv Route, aceasta din urmă reprezentând o tabelă de legătură pentru a realiza o relaţie între tabelele Lines şi respectiv, Stations, după cum se poate vedea în figura următoare:

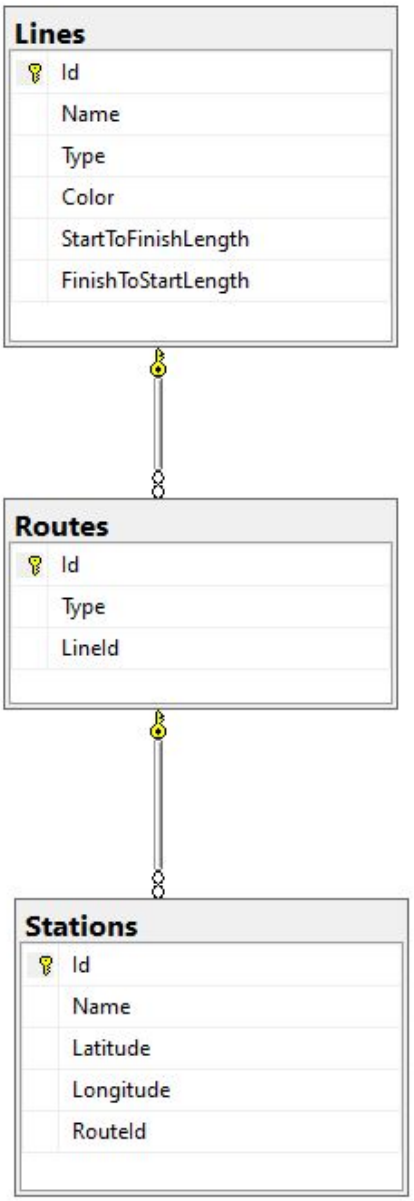


Figura 20: Schema tabelelor folosite pentru importare de date

Relaţiile între aceste tabele sunt următoarele:

- Între tabela Lines şi tabela Routes este o relaţie de tip one to many, astfel, o linie poate avea una sau mai multe rute. Această relaţie are la bază faptul că în timp ce tramvaiele au un singur traseu pe care îl parcurg, autobuzele au 2, de aici nevoia de a avea mai multe rute pentru o linie
- Între tabela Routes şi tabela Stations este o relaţie de tip one to many, deoarece o rută este compusă dintr-o listă de staţii parcurse într-o anumită ordine

Entităţile ActiveLine şi ActiveStation reprezintă adaptările entităţilor Line şi respectiv, Station. Acestea sunt folosite în cadrul aplicaţiei pentru a ţine evidenţa liniilor a căror locaţie este expusă prin intermediul locaţiei utilizatorilor.

În cazul acestor entităţi nu este necesară o tabelă de legătură deoarece o linie activă are la un moment dat un singur traseu, deci o singură rută, caz în care lista de staţii ce compun o rută pot fi referenţiate direct în cadrul liniei active.

Aceste tabele pot fi vizualizate în figura următoare:

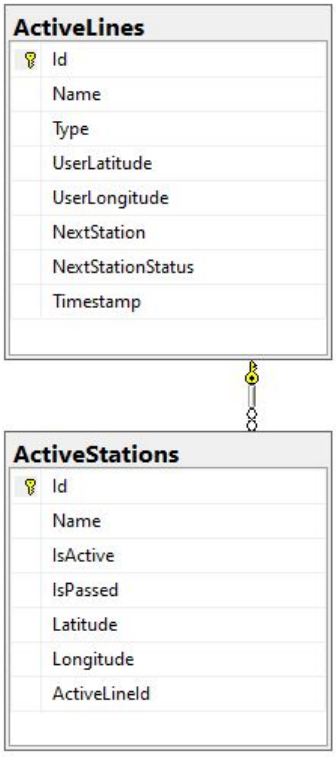


Figura 21: Schema tabelelor folosite pentru logica aplicaţiei

Relaţiile între aceste tabele sunt următoarele:

- Între tabela ActiveLines şi tabela ActiveStations este o relaţie de tip one to many

Schema completă a bazei de date poate fi vizualizată în figura următoare:

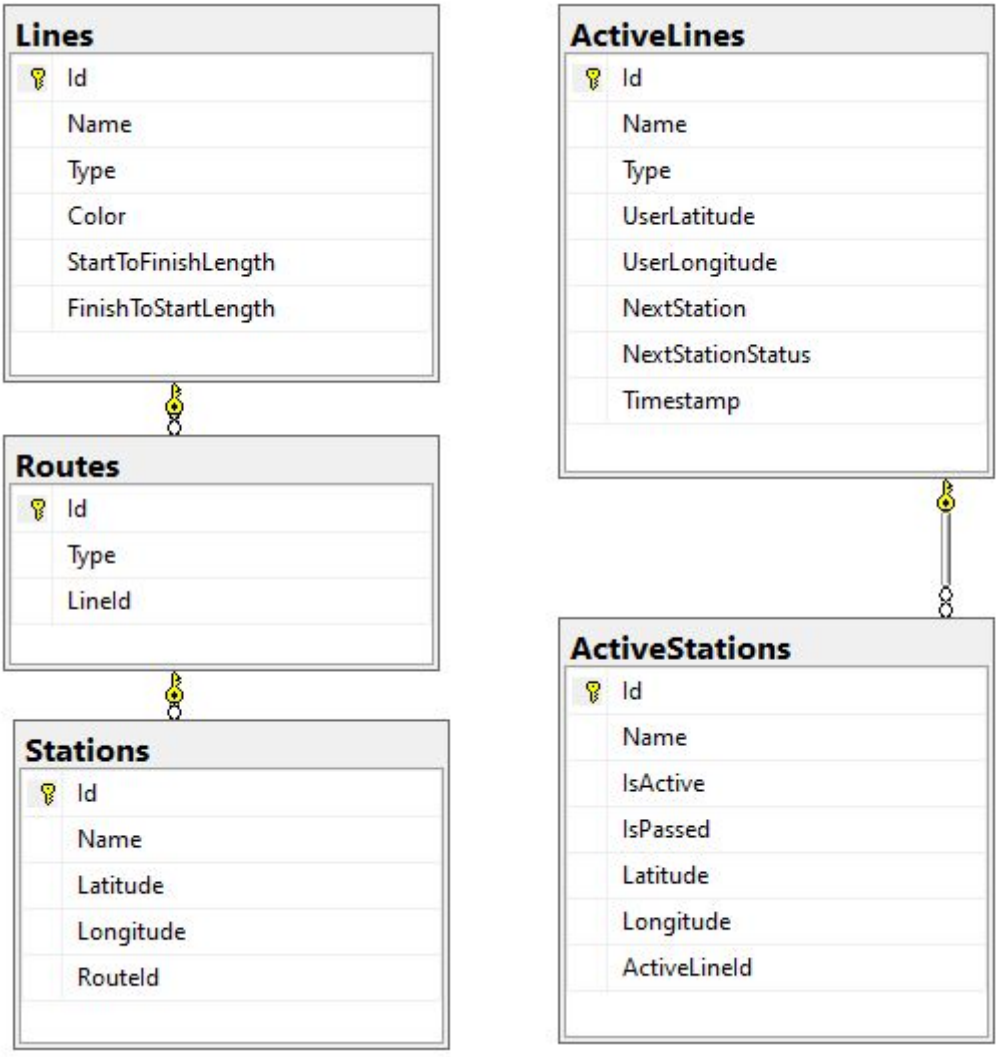


Figura 22: Schema completă a bazei de date

# CONCLUZIILE LUCRĂRII

## 1. Concluzii generale

Scopul aplicaţiei este de a facilita utilizarea serviciilor de transport public, prin a oferi utilizatorilor informaţii concrete despre locaţia liniilor ce reprezintă un interes pentru aceştia, şi nu aproximări care deşi merg spre un nivel de detaliu mai mare, cum ar fi o aproximare atât a orei cât şi a minutului în care o linie va ajunge într-o anumită staţie, dar a căror acurateţe fluctuează.

Personal, m-am găsit de numeroase ori în postura de a fi nevoit să sacrific o parte din timpul meu pentru a aloca suficient timp de aşteptat în staţie pentru a putea ajunge la destinaţie, situaţie care este cu atât mai neplăcută cu cât este mai limitat timpul pe care îl am la dispoziţie şi mai important să ajung la destinaţie la timp, de exemplu într-o zi cu examen. Acest tip de situaţie putea fi evitată dacă aş fi avut o sursă de încredere ce să ofere informaţii legate de locaţia mijloacelor de transport, astfel aş fi putut să-mi organizez timpul luând în calcul aceste informaţii.

Abordarea folosită în această aplicaţie este de a folosi locaţia utilizatorilor pentru a o putea identifica pe cea a mijloacelor de transport public cu care aceştia călătoresc. Astfel, utilizatorii se împart în 2 categorii: cei care expun locaţia liniilor prin intermediul locaţiei lor, şi cei care caută linii printre cele expuse de ceilalţi utilizatori.

În calitate de utilizator ce caută linii pentru călătoria sa, detaliile legate de linii pot fi vizualizate în 3 moduri:

- **Minimal**: toate liniile potrivite căutării sunt afişate cu următoarele detalii: numele acesteia, traseul pe care îl parcurge, următoarea staţie şi statusul liniei relativ la distanţa acesteia de staţii.
- **Detaliat**: detaliile unei singure linii sunt afişate, acestea conţinând: numele acesteia, traseul pe care îl parcurge, un număr mai mare de staţii astfel încât utilizatorul poate vedea câte staţii mai are linia de parcurs până la acesta, staţiile sunt marcate corespunzător în funcţie de distanţa acestora de linie, sau dacă deja au trecut prin respectivele staţii

- **Hartă**: toate liniile sau liniile filtrate de către utilizator pot fi vizualizate pe o hartă astfel încât utilizatorul să poată avea o privire de ansamblu a opţiunilor curente din punct de vedere al mijloacelor de transport

Dezvoltarea acestei soluţii mi-a oferit prilejul de a învăţa să construiesc o aplicaţie end to end, m-a provocat să gândesc o arhitectură ce să se îmbine atât funcţionalităţi pentru importare de date, prelucrarea acestora, cât şi extensibilitate, cuplaj mic, astfel încât aplicaţia să poată fi extinsă pe viitor la numeroase oraşe. A fost o provocare la nivel arhitectural, una din care am avut multe de învăţat şi mi-a extins viziunea din punct de vedere al abordărilor posibile pe partea de structurare a unei aplicaţii.

## 2. Direcţii de viitor

În viitor, ca şi funcţionalitate ar putea fi adăugată o pagină cu detalii despre linii ce se potrivesc cu cele mai comune căutări ale utilizatorului, cum de multe ori căutările se vor repeta, de exemplu linii pentru trasee acasă - facultate, acasă - loc de muncă, etc., aceasta ar fi o adiţie ce ar aduce valoare, deoarece l-ar privi pe utilizator de a repeta aceeaşi acţiune zilnică de căutare.

O funcţionalitate ce ar veni în completarea celei legate de cele mai comune căutări ar putea fi un sistem de notificări, care să trimită notificări utilizatorului legate de linii care de obicei sunt de interes pentru acesta care sunt în apropiere, eventual programate pe un plan pe zile ale săptămânii, de exemplu utilizatorul să poată seta în aplicaţie să primească notificări în fiecare luni la ora 10:00 legate de trasee pe care să le introducă manual sau legate de traseele cele mai comune ale utilizatorului.

De asemenea, este posibilă pe viitor este trecerea de la dependinţa de utilizatorii aplicaţiei pentru a urmări locaţia mijloacelor de transport la utilizarea datelor expuse de CTP în cazul în care acestea devin mai stabile şi mai concrete.

# BIBLIOGRAFIE


1. Angular: https://angular.io/
2. Angular Material: https://material.angular.io/
3. HTML5 Geolocation API: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API
4. Entity Framework Core: https://docs.microsoft.com/en-us/ef/core/
5. ASP.NET Core 3.0: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.0
6. AutoMapper: https://docs.automapper.org/en/stable/Getting-started.html
7. Formula Haversine: https://en.wikipedia.org/wiki/Haversine_formula
8. SSMS: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15
9. HEREITIS: http://hereitis.ro/
10. Moovit: https://moovitapp.com/
11. Google Maps: https://www.google.ro/maps
12. Google Maps API: https://angular-maps.com/api-docs/agm-core/components/agmmap


## Link-uri poze:


1. [1] https://cdn.mobilesyrup.com/wp-content/uploads/2018/05/moovitTripSearch.jpg
2. [2] https://techcrunch.com/wp-content/uploads/2017/01/google-maps-parking.png?w=730&crop=1
3. [3] https://image.winudf.com/v2/image1/Y29tLnJlYWx0aW1lLmJ1c2VzX3NjcmVlbl8wXzE1NjIwMTYwMDRfMDQ2/screen-0.jpg?fakeurl=1&type=.jpg
4. [4] https://angular.io/generated/images/guide/architecture/overview2.png

# ANEXE

## 1. Angular

Aplicaţia a fost dezvoltată la nivel de client folosind framework-ul web Angular, versiunea 8.3.20. Am ales Angular deoarece prezintă numeroase avantaje ce contribuie la productivitate şi la dezvoltare, cum ar fi:

- Linia de comandă: aceasta ajută la crearea automată de componente, servicii, testarea şi rularea aplicaţiei, etc.
- Suport: multe IDE-uri prezintă numeroase funcţionalităţi pentru Typescript, limbajul folosit în dezvoltarea aplicaţiilor Angular, cum ar fi: intellisense, code snippets, etc.

  Mai jos poate fi vizualizată arhitectura Angular.

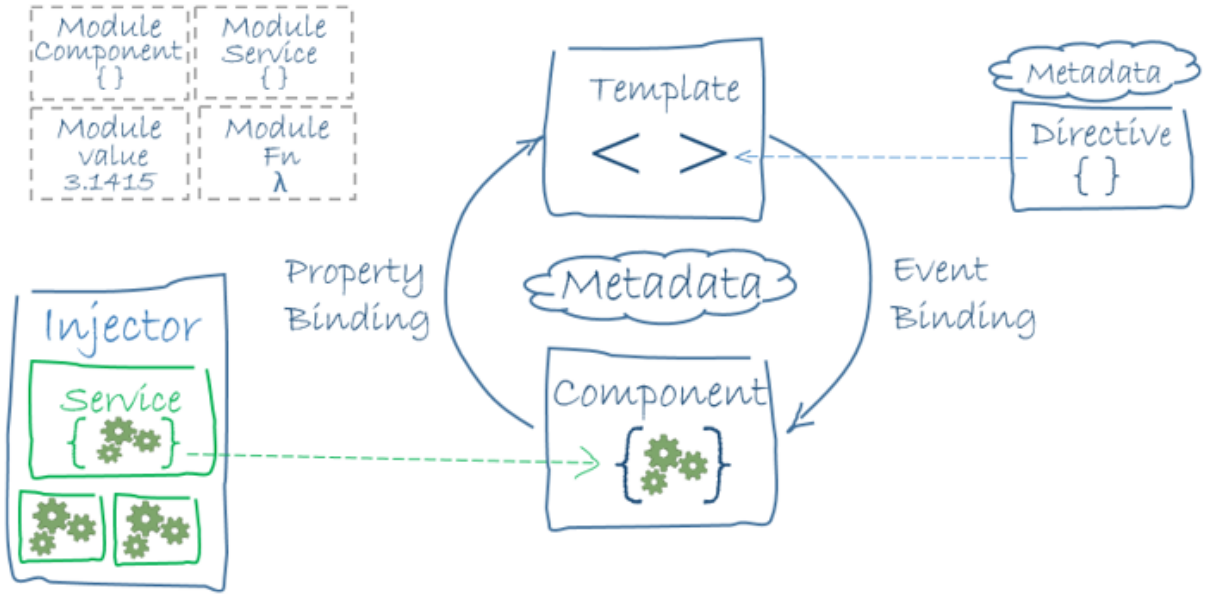


Figura 23: Arhitectura unei aplicaţii Angular [4]

După cum putem observa, o aplicaţie Angular se bazează pe componente, în cadrul cărora sunt încapsulate fişierele .html, .css, . ts şi .spec.ts pentru fiecare funcţionalitate. În aceste componente putem de asemenea injecta servicii definite de noi înşine sau servicii predefinite precum cel de rutare, Angular având suport pentru Dependency Injection.

## 2. Angular Material

Angular Material este o librărie pentru Angular ce conţine componente predefinite. Motivele pentru care am ales această librărie sunt:

- Consistenţă în interfaţă
- Uşurinţă în utilizare
- Design plăcut
- Performanţă

## 3. ASP.NET Core

Partea de server a fost construită folosind ASP.NET Core. Acesta este un framework open-source, cross-platform, modern, ce facilitează construirea de aplicaţii web end to end, API-uri, etc. Motivele pentru care am ales ASP.NET Core sunt următoarele:

- Prezintă o integrare uşoară cu framework-uri client-side
- Suport pentru Dependency Injection încorporat
- Performanţă

## 4. Entity Framework Core

Am ales Entity Framework Core pentru a manageria partea aplicației ce ține de interacţiunea cu baza de date. Motivul principal pentru care am ales această tehnologie este faptul că oferă posibilitatea de a lucra cu date prin intermediul unor clase, şi nu direct cu baza de date.

## 5. AutoMapper

AutoMapper este o librărie scrisă cu scopul de a privi dezvoltatorii de scrierea repetitivă a aceloraşi metode de mapare între obiecte. Această librărie se bazează pe convenţii, fapt ce uşurează folosirea acestuia. Am ales să folosesc AutoMapper pentru facilitarea mapărilor de la entităţi la DTO-uri, şi vice versa.

## 6. HTML5 Geolocation API

Am ales să folosesc acest HTML5 Geolocation API deoarece este integrat direct în browser, acest fapt facilitând cu mult construirea logicii în jurul acestuia. Acest API are numeroase avantaje, fapt pentru care am decis să îl folosesc în construirea aplicației. Unele dintre aceste avantaje sunt:

- Compatibilitate: fiecare browser are compatibilitate completă cu acest API
- Securitate: funcționează doar pe HTTPS, permisiunea utilizatorului este cerută înainte de a prelua orice informație legată de locația acestuia
- Este ușor de folosit, prezintă funcționalități relevante în contextul identificării/urmăririi locației.

## 7. Google Maps API

Am folosit Google Maps API pentru a implementa modul de vizualizare pe hartă. Motivele pentru care am ales acest API sunt următoarele:

- Popularitatea: Google Maps este unul din cele mai populare servicii oferite de Google, prin consecință, majoritatea utilizatorilor vor fi deja familiarizați cu interfața
- Performanța
- Stabilitatea
- Este ușor de folosit