MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Sistem de monitorizare a limitei de viteză destinat conducătorilor auto

LUCRARE DE LICENŢĂ

Absolvent: **Ilieș Alexandru**

Coordonator științific: **Șef lucrări ing. Cosmina IVAN**

2016

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Ilieș Alexandru**

Sistem de monitorizare a limitei de viteză destinat conducătorilor auto

1. **Enunţul temei:** Proiectul reprezintă o sistem care servește drept asistent pentru conducătorii auto. Aplicația este una cât se poate de practică și utilă, care își propune să ofere posibilitatea de a reduce stresul, neplăcerile care pot apărea din neatenția sau din alt motiv a conducătorului auto atunci când vine vorba de limita de viteză.
2. **Conţinutul lucrării:** Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare, Validare și Evaluare, Manual de Instalare și Utilizare, Concluzii, Bibliografie, Anexe.

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**: Șef lucrări ing. Cosmina IVAN

5. **Data emiterii temei:** 1 noiembrie 2015

6. **Data predării:** _____

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a)_____
_____,
legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării
_____
_____
_____elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

_____

Nume, Prenume

_____

Semnătura

Cuprins

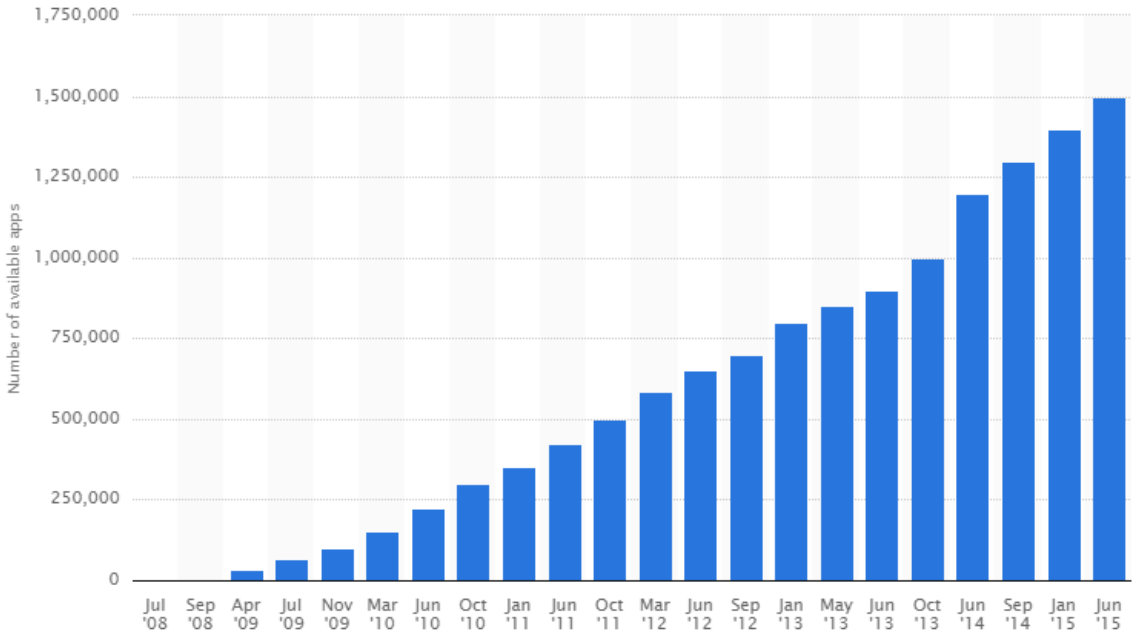
Capitolul 1. Introducere – Contextul proiectului

În ultimii douăzeci de ani cu toții suntem martori ai unei evoluții uimitoare ce privește tehnologia informațională. Cum se descrie și în articolul [1], evoluția tehnologiilor, cât și apariția diferitor dispozitive, cum ar fi: calculatoarele, diferite echipamente periferice, dispozitive mobile(senzori, telefoane inteligente, gadget-uri, etc.) oferă un suport de neînlocuit cerințelor umane. Toate aceste tehnologii, dispozitive fizice oferă posibilitatea de creare a aplicațiilor pentru toate tipurile de cerințe, din orice domeniu al vieții. Având la îndemână dispozitivele sau gadget-urile necesare, acestea servesc ca o adevărată armă cu care se poate lupta pentru a spori productivitatea în diferitele cerințe de business, pentru a spori procesul de educație, pentru a deveni mai buni în ceea ce facem și de a evolua continuu.

1.1. Contextul proiectului

Tehnologiile lumii mobile cât și dispozitivele mobile, în momentul de față sunt prezente în toate ramurile societății. La fel cum se descrie și în sursa [4], dispozitivele mobile sunt folosite aproximativ peste tot, în orice domeniu, făcând viața mai ușoară prin modul în care ajută oamenii să-și îndeplinească cerințele sau nevoile atât personale cât și cele legate de business. Chiar dacă la începutul erei dispozitivelor mobile, mai exact al telefoanelor inteligente, toate acestea erau privite cu scepticism (în afară de faptul că făceau comunicarea mult mai ușoară), astăzi situația este cu totul diferită. Telefoanele inteligente servesc în mod clar la ușurarea activităților umane și nu numai. Din cauza diferitor posibilități care apar folosind un dispozitiv inteligent, din cauza performanțelor la care acestea au ajuns la ora actuală, cât și din cauză că internetul a împânzit în totalitate această ramură acestea sunt din ce în ce mai folosite. Dacă acum 10 ani era necesar strict un PC pentru a îndeplini sarcinile de care ai nevoie, astăzi această restricție este eliminată deoarece dispozitivele mobile sunt cu mult mai practice și destul de performante pentru a îndeplini acele sarcini.

Aici se adaugă și inteligența cu care pot fi dotate aceste dispozitive. Diferiți senzori, diferite aplicații – fac ca posibilitățile și popularitatea folosirii acelui dispozitiv să crească din ce în ce mai mult. Un dispozitiv mobil, cum este un telefon inteligent – schimbă modul în care oamenii comunică, lucrează, evoluează și se distrează. Acestea toate se datorează nu în ultimul rând aplicațiilor care sunt disponibile și se pot descărca pe acest telefon. Aplicațiile mobile sunt în continuă creștere. În figura 1.1 se observă clar sporirea numărului de aplicații și tendința cu care acestea se dezvoltă, iar acestea sunt numai aplicații mobile.



Figură 1.1 Număr aplicații mobile disponibile începând cu luna iulie 2008 - iunie 2015

Aplicațiile mobile sunt create pentru toate ramurile vieții. Există aplicații care îți monitorizează sănătatea, altele te ajută să fii mai productiv în domeniul educației, în domeniul afacerilor, iar altele te ajută să evoluezi în viața de zi cu zi.

Toate aceste fenomene, cum sunt: dezvoltarea tehnologiilor, folosirea pe scară largă a dispozitivelor mobile și nu în ultimul rând al aplicațiilor mobile – îi motivează pe dezvoltatori să aducă mereu pe piață noi aplicații, care sunt mult mai evoluate, mature, mult mai interactive.

Aplicația prezentată aici, este una mobilă, care face parte din categoria: educațional-productivă, destinată în primul și în primul rând conducătorilor de autoturisme. La conducătorii auto, mai ales la conducătorii începători, se depistează un procentaj de stres mărit. Acesta este provocat în mare parte de experiența precară pe care conducătorul o are. Din acest motiv, de multe ori se întâmplă să fie neatenți mai ales când vine vorba de indicatoarele de circulație. Este cunoscut faptul că, chiar și conducătorii experimentați de multe ori se întâmplă să fie neatenți. Motivele pot fi diferite, iar pentru asta aplicația realizată are ca scop ajutorarea conducătorului auto atunci când vine vorba de limita de viteză. Aici merită de evidențiat mai multe motive pentru care limita de viteză devine o problemă pentru conducătorii auto.

Următoarele situații pot duce la unele neplăceri mai puțin sau mai mult serioase conducătorului auto:

- Conducătorul auto din neatenție a depășit limita de viteză, iar pentru asta a fost amendat.
- Conducătorul auto a fost destul de atent însă din mai multe motive posibile nu a observat indicatorul de viteză (acesta poate să lipsească, mai rar dar este posibil) iar aceasta poate duce din nou la sancțiuni.
- Conducătorul auto este învinuit că a depășit limita de viteză, și totuși acesta nu este sigur că a depășit limita de viteză și dorește mai multe detalii.

Mai sus sunt descrise motivele și problemele esențiale care pot să apară atunci când se pune problema limitei de viteză. Soluția care este implementată în aplicație este următoarea: la un interval de timp predefinit (1/1,5 secunde) aplicația primește informații precum: numele/denumirea străzii/segmentului de drum pe care automobilul se află și viteza maximă permisă pe acel segment de drum. Având aceste date, în funcție de viteza de deplasare a automobilului, aplicația comunică cu un server unde transmite date de genul: locația, viteza de deplasare, viteza maximă permisă pe acel segment de drum. Pe lângă cele spuse, aplicația poate servi drept auto-educare pentru conducătorii auto. Una dintre funcționalitățile aplicației, oferă posibilitatea de a salva un istoric al încălcărilor și de a face anumite statistici a acestora. Pe baza acestei analize utilizatorul aplicației își poate vedea corectitudinea sa în legătură cu încălcarea limitei de viteză.
Pe lângă aplicația care este descrisă mai sus, proiectul de față are ca scop și accentuarea unor tehnologii mai noi, care cresc din ce în ce mai mult în popularitate, diferite de cele standard de implementarea a aplicațiilor mobile cum sunt: Objective - C/Swift pentru dispozitivele inteligente care rulează iOS, Android – pentru dispozitivele care rulează Android. Windows Mobile – pentru dispozitivele care rulează Windows. Aceste tehnologii reduc timpul necesar implementării aplicațiilor, reduc totodată costurile necesare dezvoltării aplicațiilor și nu în ultimul rând – munca necesară dezvoltării aplicațiilor.

1.1.1. Motivația

Ținând cont de problemele cu care se pot confrunta șoferii în legătură cu depășirea limitei de viteză, cât și de unele cazuri mai speciale, conform studiului prezent în sursa [2], unde au fost întrebați un număr de 50 de șoferi care au fost implicați într-un accident, doar 33% cunoșteau exact limita de viteză. Dintre aceștia, aproape 90% respectau limita de viteză cunoscută. De-aici rezultă că 60% din cei 50 interogați sunt ignoranți referitor la limita de viteză. Aici pot fi diferite motive. De exemplu sunt cazuri când un indicator de viteză din oricare motiv lipsește sau din neatenție nu a fost observat.

Datorită motivelor de mai sus, a apărut ideea de implementare a acestei aplicații care va face ca apariția acestor probleme să fie redusă la minim. Un caz concret unde apare astfel de probleme sunt autostrăzile pe care nu există o limită de viteză, și anume: fiind dată o autostradă în care există porțiuni de drum pe care nu se înregistrează o limită de viteză, în sens că se poate accelera oricât se dorește, poți omite din neatenție indicatorul de viteză. Asta se întâmplă din motivul că viteza de deplasare este foarte mare, iar atenția conducătorului auto este concentrată la maxim și probabilitatea de a observa indicatorul este redusă. Toate cele descrise mai sus, au contribuit la motivarea și la dezvoltarea unei aplicații care servesc la rezolvarea acestor tipuri de situații. Utilizarea unei astfel de aplicații, oferă avantaje pentru aproximativ toți conducătorii auto.

1.2. Conținutul lucrării

Acest document are următoarea structură:

Capitolul 1 (Introducere)

În acest capitol se prezintă detalii generale despre starea actuală de clasificare a aplicațiilor mobile. Sunt reprezentate motivele și necesitatea implementării unui astfel de sistem și ce probleme acesta rezolvă.

Capitolul 2 (Obiectivele Proiectului)

În acest capitol se prezintă obiectivele propriu-zise ale sistemului, prezentate în detalii.

Capitolul 3 (Studiu Bibliografic)

În acest capitol sunt prezentate tehnologiile existente, sisteme existente și diferite comparații între acestea. Datorită acestor cercetări s-a optat pentru anumite tehnologii împreună cu motivarea alegerii.

Capitolul 4 (Analiză și Fundamentare Teoretică)

Acest capitol cuprinde explicații precum principii de funcționare ale sistemului, principii de funcționare a tehnologiilor folosite cu scopul atingerii obiectivelor sistemului și implementarea cerințelor funcționale și non-funcționale.

Capitolul 5 (Proiectare de Detaliu și Implementare)

În acest capitol se explică în detaliu cum a fost proiectat și cum funcționează sistemul. Sunt prezentate componentele care fac parte din sistem și modul cum acestea interacționează între ele.

Capitolul 6 (Testare și Validare)

Acest capitol cuprinde partea de testare și de validare a sistemului. Sunt explicate metodele care au fost aplicate pentru testarea sistemului.

Capitolul 7 (Manual de Instalare și Utilizare)

În acest capitol se explică modalitățile prin care sistemul poate fi instalat și instrucțiunile de folosire.

Capitolul 8 (Concluzii)

Acest capitol conține o analiză împreună cu un set de concluzii legate de sistem și de modul în care acesta poate fi îmbunătățit sau dezvoltat.

Capitolul 2. Obiectivele Proiectului

2.1. Obiectivele sistemului

În proiectul de față scopul principal este crearea unui sistem, care va rula pe dispozitive mobile, destinat în mare parte conducătorilor auto. Dispozitivele mobile actuale sunt proiectate pentru platforme diferite, de aceea se pune accent și pe realizarea și rularea sistemului pe toate platformele existente, datorită tehnologiei care se folosește pentru implementarea acestui sistem. Acest obiectiv este foarte important, întrucât platformele existente sunt cât se poate de răspândite iar din aceasta rezultă că utilizatorii nu trebuie să se simtă intimidați din cauză că acesta nu folosește o anumită platformă. Se dorește o implementare cât mai eficientă în ceea ce privește funcționalitățile care sunt realizate și o utilizabilitate cât mai practică și comodă a aplicației. Se pune accent pe simplitatea aplicației întrucât conducătorii auto, mai ales persoanele în vârstă nu sunt foarte familiarizați cu noile tehnologii.

Obiectivele principale ale sistemului sunt următoarele:

- **Detectarea vitezei**

Acest scop permite detectarea vitezei de deplasare a vehiculului cu o acuratețe maximă posibilă. Acest scop este realizabil datorită sistemului GPS care se regăsește în majoritatea telefoanelor mobile. S-a dovedit prin testare, conform sursei [12] că acuratețea de detectare a vitezei de deplasare prin acest mod – este la fel de bună precum acuratețea unui vitezometru fizic, aflat la bordul mașinii. Se întâmplă cazuri când vitezometrele fizice la anumite vehicule se defectează, iar un bun înlocuitor sunt chiar aceste sisteme - care folosesc GPS[1]-ul pentru detectarea vitezei.

- **Detectarea vitezei maxime admise și locația**

Datorită realizării acestui scop se poate afla în orice moment viteza maximă posibilă pe segmentul de drum în care vehiculul se află, inclusiv și locația. Datorită acestor posibilități, conducătorul vehiculului poate deveni mai atent având la îndemână aplicația. Un motiv ar fi cunoașterea în orice moment, a depășirii limitei de viteză.

- **Atenționarea conducătorului auto când a încălcat limita de viteză**

Scopul prevede atenționarea conducătorului auto cu un semnal sau o notificare, precum că a depășit limita de viteză. Acest obiectiv este foarte important. Datorită unui asemenea criteriu, conducătorul autoturismului poate fi mai atent fiind conștient că a încălcat limita de viteză .

- **Păstrarea unui istoric al încălcărilor**

Realizarea acestui obiectiv permite utilizatorului o modalitatea de vizualizare a unui istoric al depășirilor limitei de viteză. Acesta conține informații precum: viteza de deplasare la momentul când s-a încălcat limita de viteză, data, ora și locația unde a avut

[1] Sistem global de navigație prin satelit și unde radio.

loc depășirea limitei de viteză. Acesta îi permite utilizatorului să vadă statistici precum numărul de încălcări, locațiile unde cel mai des a depășit limita de viteză.

- **Primire Mail la fiecare sfârșit de săptămână cu statistici**

Realizarea obiectivului dat prevede trimiterea unui mail către toți utilizatorii, care va conține o statistică a încălcărilor ce au avut loc timp de o săptămână. Această posibilitate ar fi foarte utilă pentru utilizatori. Aceștia vor putea să-și analizeze constant modul în care conduc.

Capitolul 3. Studiu Bibliografic

3.1. Aplicații mobile native vs. aplicații mobile hibride vs. aplicații web

Pentru o înțelegere mai bună a acestor trei tipuri de aplicații se va explica în detaliu, în rândurile ce urmează.

Conform sursei [5], în contextul de dezvoltare a aplicațiilor mobile, sunt trei categorii – native, hibride și aplicația web. Din acest motiv, există și dilema – pe care dintre categorii să o folosim? La dezvoltarea de aplicații mobile există 5 provocări comune pentru cele trei categorii enumerate mai sus:

Standardele multiple.

La dezvoltarea aplicațiilor este dificil să ții cont numai de un standard, din motiv că există sau cunoaștem atâtea standarde și platforme. De exemplu, pentru dispozitivele pe care rulează Android[2] există diferite specificații hardware, cum ar fi RAM, CPU[3], mărimea ecranului și capacitatea de stocare a informației. La fel aceeași problemă o întâlnim și la dispozitivele care rulează pe iOS[4].

Lipsa instrumentelor de testare.

Această provocare vine odată cu faptul că aplicațiile mobile și dezvoltarea lor au o limită de suport pentru testările automate în aplicațiile native. De exemplu, testarea senzorilor serviciului de localizare - în acest context este necesar o dezvoltare mai bună a uneltelor de analiză precum și monitorizarea diferitor metrici ale aplicației și testării.

Versiuni împărțite.

Spre exemplu pe Android, care este un sistem de operare cu sursa codului deschisă (open-source), dezvoltatorii pot modifica codul sursă și ca rezultat pot primi standarde multiple sau mai bine zis versiuni multiple ale platformei. De-aici pot ruzulta mai multe probleme legate în mare parte de compatibilități. În comparație cu Android, platforma iOS și Windows nu au sursa codului deschisă, prin urmare nu există o problemă în contextul compatibilității sistemului de operare.

Lansări frecvente de versiuni noi

Pentru fiecare sistem de operare se lansează frecvent noi versiuni, cu scopul de a fixa erori și de a securiza acest sistem de operare. (vezi tabelul 3.1). Acest fapt creează provocări dezvoltatorilor. Ei sunt puși în situația de a învăța diferite limbaje de programare pentru fiecare platformă. Adițional, ar trebui să fie puși la curent cu tot ce este nou sau orice modificare relevantă și care i-ar ajuta să fie mult mai eficienți.

[2] Sistem de operare oferit de Google
[3] Componente fizice ale unui dispozitiv
[4] Sistem de operare pentru dispozitivele mobile oferi de Apple

Anul	Android	iOS versiune
2007		1.0 1.1
2008		2.0 2.1 2.2
2009	- Banana Bread - Cucpcake - Donut - Eclair	3.0 3.1
2010	- Froyo - Gingerbread	3.2 4.0
2011	- Honeycomb - Ice cream sandwich	4.3 5.0
2012	-Jelly bean	6.0
2013	-Kitkat	7.0
2014	-Lollipop	7.1 8.0 8.1
2015	-Marshmallow	8.2 8.3 8.4 9.0 9.1
2016	N(Dev Preview 4)	9.2 9.3 10.0(Beta)

	Android	iOS
Language	Java	Objective-C/Swift
Tools	Android SDK	Xcode
Format	.apk	.ipa

Tabel 3.1 Apariția versiunilor sistemelor de operare Android și iOS

Spațiul de memorie limitat

Majoritatea dintre telefoanele mobile au o stocare de memorie limitată și în cele mai multe cazuri, acest spațiu este utilizat la maximum. Utilizând conexiunile de internet, putem stoca informațiile (data) pe serverele distribuite, în acel moment capacitatea memoriei nu mai este un subiect decisiv sau urgent de rezolvat. Dar, la un punct anumit, aplicația va avea nevoie ca informația pe care o deține să fie stocată exact în memoria telefonului, când telefonul nu este conectat la Internet.

Comparări între tipurile de aplicații.

Aplicațiile mobile native

Aplicațiile mobile native sunt instalate direct pe telefonul mobil și pot fi lansate fără unelte intermediare. Aceste aplicații pot accesa sistemul de operare a telefonului și toate serviciile oferite de către acesta împreună cu componentele fizice ale dispozitivului, cum ar fi camera, accelerometru sau GPS-ul. Oricum, dezvoltarea de aplicații native cer cunoștințe aprofundate despre o platformă specifică.

Când construim aplicația, putem avea acces direct la serviciile native care ne va oferi permisiunea să utilizăm beneficiile de soft și hard a unui dispozitiv specific. Pe de altă parte, ne putem lovi și de mari dezavantaje în dezvoltarea de aplicații mobile native, cum ar fi: sursa codului scris pentru o platformă nu poate fi utilizată pe alta.

Ca să se înțeleagă mai bine - codul sursă scris pentru Android nu poate fi utilizat pe platforma iOS. Modelul pentru aplicația nativă cere de asemenea și timp și costuri ineficiente, având în vedere și efortul depus la dezvoltarea și mentenanța acestuia. Un alt dezavantaj legat de construirea acestui tip de aplicații este că fiecare platformă cere utilitare specifice, cum ar fi SDK[5]-ul și astfel am fi forțați să utilizăm uneltele lor unice. Aplicația trebuie să fie distribuită prin magazinul aplicației, avantajul în acest caz este că nu trebuie să distribuim aplicația direct de către noi spre utilizatori. Deși, distribuirea în acest magazin necesită mii de reguli și politici de siguranță până ca aplicația să fie aprobată și încărcată pe stoc.

Aplicații WEB

Există două forme de aplicații web, una dintre ele este atunci când paginile sunt primite de către partea de server și atunci paginile sunt afișate doar de navigatorul web[6] local al utilizatorului iar o altă abordare este redarea paginilor direct din navigatorul web local. Ambele căi utilizează HTML, CSS, JavaScript [7]pentru a construi aplicația. Indiferent de abordare, aplicația web este accesată prin navigatorul web al dispozitivului. Avantajul utilizării acestei modalități de creare a aplicației ține de mica complexitate, este mai ieftin și mai rapid se va construi și nu în ultimul rând mai ușor se va menține. Găzduirea codului sursă pe un server distribuit oferă dezvoltatorilor posibilitatea de a actualiza continuu aplicația. Un alt avantaj este că partea de suport nu necesită externalități sau mai exact platforma este independentă.

Singura cerință este să se aibă la îndemână un navigator web. Trecând la capitolul dezavantaje, pot fi atunci când dispozitivul de pe care se accesează aplicația, are nevoie să funcționeze offline[8] – fără conexiune de Internet. În acest caz, dacă lipsește conexiunea, utilizatorul nu poate interacționa cu aplicația sau să acceseze oricare conținut, pentru că conținutul este stocat pe server. La fel un dezavantaj îl prezintă și serviciile native și alte caracteristici ale telefonului care nu pot fi accesate prin aplicațiile web.

Aplicații hibride.

[5] Software Development Kit sau Trusă de dezvoltare a programelor
[6] Browsers(din engleză) cum ar fi, IE, Firefox, Google Chrome
[7] Limbaje de programare și scriptare folosit pe partea de web.
[8] Fără conexiune la Internet

Aplicațiile mobile de tip hibrid combină ambele opțiuni, adică atât dezvoltare nativă cât și tehnologiile web. Aplicațiile hibrid arată similar cu cele web, din motiv că este utilizată tehnologia web pentru a prezenta conținutul, dar aplicația este folosită într-un un container nativ. Utilizând acel container aplicația poate folosi toate caracteristicile native pe care un telefon le poate oferi. Utilizând această abordare, se salvează timp și cost pentru mentenanță, pentru că codul poate fi reutilizat pe alte platforme. Spre exemplu, utilizare unui framework[9] de la o aplicație hibridă, cum ar fi Cordova[10], care este un framework open source. Cordova[11] are o interfață de tip JavaScript implementată pentru a accesa caracteristicile native ale telefonului.

Dezavantajul dezvoltării unei aplicații hibride este imposibilitatea dezvoltării independente față de OS și are nevoie neapărat de un container nativ. Acest container, oferă acces la caracteristicile native. Și cu toate acestea ultima interfață a utilizatorului riscă să nu fie suportată în totalitate. Performanța nativă nu poate fi dobândită cu o aplicație hibridă, odată ce a fost aceasta rulează într-un container și utilizează navigatorul web al dispozitivului pentru a prezenta sau a reda conținutul. Cele mai importante platforme cu care se pot crea aplicații hibride sunt prezentate în următorul tabelul 2.2.

Nume	**Platforme suportate**	**Disponibilitate în magazinele de aplicații**
WebWorks	BlackBerry 5.x-7.x BlackBerry PlayBook BlackBerry 10	BlackBerry AppWorld
Windows 8 Store	Windows 8 Store	Windows 8 Store
Nokia S40 webapps	Nokia Series 40	Nokia Store
Symbian webapps	Symbian webapps	Symbian webapps
Mozilla Open Web Apps	Mozilla Open Web Apps	Mozilla Open Web Apps
Chrome Packaged Apps	Desktop, Chrome OS Android	Chrome Store
Apache Cordova / Adobe PhoneGap	iOS Android BlackBerry Windows Phone Bada / Tizen Symbian	Apple AppStore Google Play Store BlackBerry AppWorld Microsoft Windows Phone Store Nokia Store

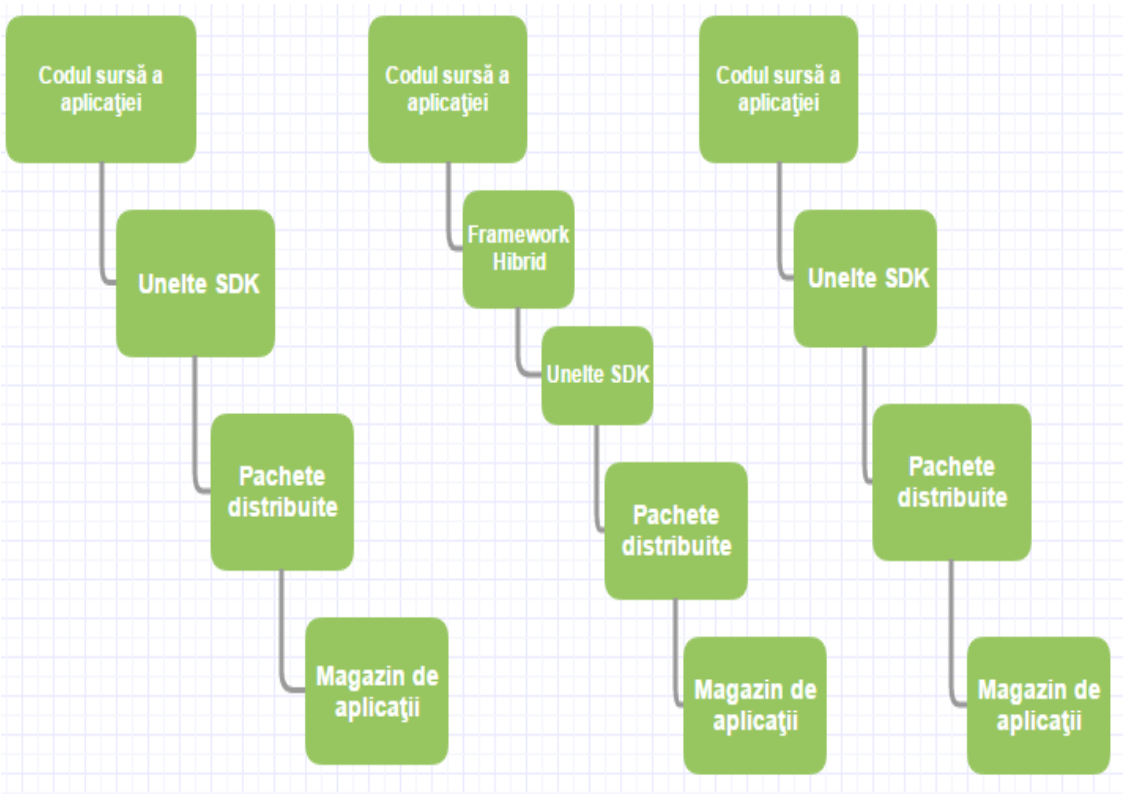
Tabel 3.2 Comparări între platformele de dezvoltare a aplicațiilor hibride

[9] Un ansamblu standardizat de concepte, practici și criterii.
[10] Framework pentru dezvoltarea aplicațiilor hibride.
[11] Framework de dezvoltare a aplicațiilor hibride

Concluzii

Având în vedere perioada scurtă cu care se poate realiza o aplicație hibridă, s-a optat pentru această metodă de realizare a aplicației. S-a ales abordarea hibridă, pentru că se dorește să se urmez principiul *build once, deploy anywhere*. Așadar, dezvoltarea unei aplicații hibride cu tehnologii web oferă altor utilizatori care dețin orice dispozitiv și orice platformă să poată folosi aplicația dată.

Dacă se dorește realizarea unei aplicații care este necesar să proceseze un flux mai mare de date local, sau dacă dispozitivele pe care se dorește a fi rulată aplicația sunt mai vechi și nu au caracteristici fizice actuale – atunci din start se poate opta pentru aplicațiile native.



(a) Aplicații native (b) Aplicații hibrid (c) Aplicații Web

Figură 3.1 Abordări posibile de dezvoltare a aplicațiilor mobile.

3.2. Nokia HERE Maps Rest API vs Google Maps Rest API

Pentru realizarea sistemului a fost necesar să fie folosite serviciile unor companii specializate în cartografierea digitală a globului pământesc. Sunt cunoscuţi mai mulţi furnizori mari de aceste servicii. Dintre cele mai populare și mai mature servicii se găsesc următoarele: Bing Maps, Here Map by Nokia, WikiMapia Map, MapQuest Map, TomTom Map, Wase Map, Yandex Map. Dintre aceste servicii, servicii care oferă metainformaţii despre un anumit segment de drum, cum ar fi cea de limită de viteză sunt Goole Maps şi HERE Maps.

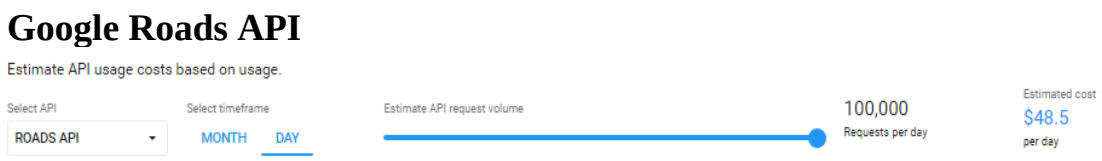
Nokia HERE Rest API	Caracteristici
Map Tile API	Returnează harta sub formă de imagini, cum ar fi hartă normală, imagini satelitare şi imgini de teren
Map Image API	Oferă imagini rapide, cum ar fi imagini pre-randate
Venue Maps API	**Venue Images:** Se pot vedea mii de locuri folosind imagini în format png/js **Venue Model:** Se pot interacţiona cu mii de modele JSON
Routing API	**Direcţii ale şoferilor şi pietonilor:** Oferă direcţia de conducere şi de mers pe jos, cu instrucţiuni în mai multe limbi. Utilizatorii pot seta preferinţe, cum ar fi cel mai scurt, sau cel mai rapid traseu, restricţii, taxe de trecere pe autostrăzi şi altele. **Traficul rutelor activat:** oferă informaţii real-time despre rutile cerute. Traficul rutelor pentru pietoni: oferă direcţiile pentru pietoni, combinate cu rute alternative, indormaţi despre străzi şi altele. **Rute pentru camion:** direcţia rutelor şi calcularea timpului de deplasare folosind atributele camionului. Suportă toate restricţiile fizice cum ar fi: greutate, înălţimea şi altele
Geocode API	**Geociding**: converteste adresele in ge-coordinate Reverse Geocoding: returnează descrierea unei locaţii folosind coordonatele geografice **Multi Reverse Geocoding:** pertime încărcarea a maxim 100 de perechi de coordonate geografice, care sunt procesate într-un singur request sincron Batch Geocoding: permite utilizatorului să încarce mai multe coordonate geografie folosind un singur fişier.
Place API	Oferă posibilitate de căutare şi descoperirea numelor, adreselor, categoriilor şi informaţiile de contact a punctelor care sunt căutate.
Trafic API	Asigură fluxul de trafic şi harta cu informaţii despre trafic.
Wheater API	Oferă informaţii despre vreme, şi alerte pentru o locaţie specifică.
Transit API	Oferă cele mai bune rute, inclunzând şi direcţii de

	mers şi staţii.
Extinderi ale platformei	**Caracteristici**
Custom Locaţion Extenion API	Salvare, administrare şi accesarea punctelor de interes şi a poligoanelor
Platform Data Extension API	Oferă conţinut adiţional, cum ar fi valorile de înălţime şi de pantă, curburi, limite de viteze şi semafoare
Route Match Extension API	Selectarea routelor GPS
Toll Cost Extension API	Calcul cost rute

Google Maps APIs Web Services	**Descriere**
Google Maps Directions API	Calculează direcţiile dintre locatii. Se poate căuta direcţiile in câteva moduri de transportare, incluzând tranzitul, mersul, sau ciclismul.
Google Maps Distance Matric API	Este un serviciu ce oferă distanţa de deplasare pentru o matrice de origini şi destinaţii, bazată pe traseul recomandat între punctele de început şi sfârşit
Google Maps Elevation API	Furnizează date de elevaţie pentru toate locaţiile de pe suprafaţa pământului, inclusiv locaţii de adâncime de pe fundul oceanului.
Google Maps Geocoding API	Serviciu ce furnizează coordonatele geografice ale adreselor şi invers
Google Maps Geolocation API	Returnează locaţia şi acurateţea cu care se depistează locaţia.
Google Maps Roads API	Indentifică informaţii despre drumurile pe care un anumit vehicol poate traversa şi furnizează totodată şi informaţii adiţionale precum metadate despre drumuri, cum ar fi **limita de viteză**.
Google Maps Time Zone API	Furnizează date despre timp în diferite locaţii de pe suprafaţa pământului.
Google Places API	Informaţii updatate la zi despre milioane de locaţii

Figura 3.3 Descrierea serviciilor oferite de Google şi Here.

Servicii echivalente ale celor două platforme, care oferă limita de viteză pentru un segment de drum, sunt : Google Roads API, şi Platform Data Extension API. Problema care apare este – costul. Nu toate aceste servicii sunt oferite gratuit. Mai jos sunt detalii despre costurile aplicate pentru permiterea folosirii acestor servicii.



Sursă https://developers.google.com/maps/pricing-and-plans/

HERE API

currency EUR €	Evaluation 0 € 90-day trial of entire platform Try for free	Starter 49 € monthly or 490 € per year with annual billing Sign up	Standard 99 € monthly or 990 € per year with annual billing Sign up	Advanced 299 € monthly or 2 990 € per year with annual billing Sign up
Transactions per Month What is a transaction?	100,000	150,000	275,000	700,000
Map & Satellite Tiles	✓	✓	✓	✓
Map & Satellite Images	✓	✓	✓	✓
Geocoding	✓	✓	✓	✓
Reverse Geocoding	✓	✓	✓	✓
Multi Reverse Geocoding	✓	✓	✓	✓
Places	✓	✓	✓	✓
Car & Pedestrian Routing	✓	✓	✓	✓
Traffic Tiles	✓		✓	✓
Public Transit Routing	✓		✓	✓
Traffic Enabled Routing	✓			✓
Traffic Incidents	✓			✓
Street Level Imagery	✓			✓
Venue Images	✓			✓
Venue Models *	✓			
Truck Routing *	✓			
Isoline Routing *	✓			

Sursă: https://developer.here.com/plans/api/consumer-mapping

Figură 3.2 Comparație costuri Google Maps vs HERE APIs.

După cum se poate vedea, costurile sunt diferite. Diferența dintre serviciul oferit de Google este că acesta nu oferă o perioadă de evaluare în care se poate testa acest serviciu. De aceea s-a optat pentru folosirea unei alternative, aceasta fiind serviciu oferit de HERE Maps.

3.3. HERE Maps API

Pentru integrarea serviciului dat şi accesarea de metadate despre segmentele de drum, HERE pune la dispozitie diferite servicii web (Application Programming

Interface) care se bazează pe standardul de comunicare JSON.[12] Acest API[13] se folosește în următorul mod:
Înainte de toate se creează un cont de evaluare și se obține API-key-ul care se creează la înregistrarea aplicației la adresa https://developer.here.com/plans/api/consumer-mapping.
Un exemplu de folosire a unui serviciu, este următorul: selectarea locației pe baza unei adrese date.

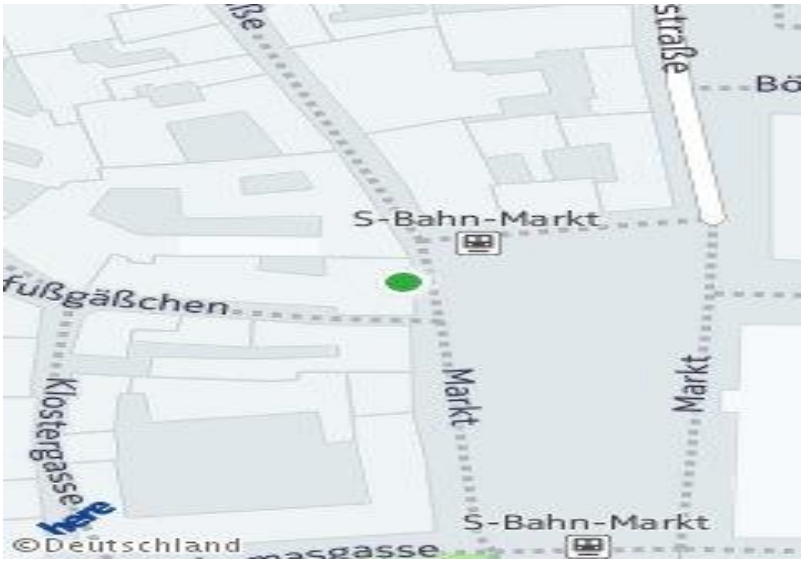
Caz de utilizare: utilizatorul dorește să obțină o hartă vizuală bazându-se pe niște parametri de import.

Se poate specifica o poziție folosind o anumită adresă în loc să se folosească coordonatele geografice(latitudine, longitudine). Următorul exemplu folosește următorii parametri: **ci=leipzig, zi=04109, s=markt** și **n=9** pentru a seta țara, orașul, codul poștal, strada și respectiv numărul casei.

Cerere:

```
https://image.maps.cit.api.here.com/mia/1.6/mapview
?app_id={YOUR_APP_ID}
&app_code={YOUR_APP_CODE}
&co=germany
&ci=leipzig
&zi=04109
&s=markt
&n=9
&z=17
&h=320
&f=1
```

Răspuns:



Figură 3.3 Locație Leipzig, Germania, specificată după adresa străzii.

[12] Format standard de reprezentare și interschimb de date între aplicații informatice
[13] Serviciu web

3.4. Sisteme similare

Înainte de începerea implementării acestei aplicații s-a urmărit un studiu de piață care a prevăzut depistarea sistemelor similare care deja există, ce au ele deja implementat și ce se poate implementa mai mult decât atât. Rezultatele au arătat că existe câteva sisteme similare, iar aceste sisteme sunt descrie mai jos.

Waze

Conform informațiilor găsite în sursa [24], Waze este o aplicație de navigație GPS revoluționară ce se adresează utilizatorilor telefoanelor mobile inteligente (smartphone). Spre deosebire de alte aplicații de navigație, avantajul aplicației Waze este acela că te va duce la destinație pe calea cea mai rapidă, ținând cont de valorile din trafic în timp real. Waze este, cel puțin în acest moment, alternativa la TM[14]C (Traffic Message Channel), dar cum în România nu există așa ceva, este singura aplicație care îți va furniza date din trafic în timp real, iar rutele tale vor fi optimizate în permanență în funcție de actualizările ce vin în timp real din trafic." Waze poate rula pe aproape orice telefon smartphone**:** Android, iPhone, Windows Mobile, Symbian si chiar pe BlackBerry. Waze necesită o conexiune permanentă la internet pentru a putea rula. Waze este Gratuit. Nu există abonament, totul fiind 100% gratuit. Singurele costuri implicate reprezintă traficul de date, iar, într-o lună de zile, Waze consumă câțiva zeci de mega, în funcție de cât de intens este folosit. Hărțile GPS din Waze sunt create de utilizatori.

Hărțile se actualizează destul de frecvent (uneori o dată pe lună, alteori o dată pe săptămână).

Cum funcționează? Aplicația Waze trimite în mod automat valorile de trafic (viteza medie pe segmente de drum) și aceasta este primită automat de toți utilizatorii Waze, evidențiindu-se chiar pe hartă zonele cu probleme, iar aplicația te va redirecționa pentru a evita ambuteiajele..

Studiul privind sistemul Waze a fost realizat înainte de data de 28 martie 2016. Începând cu 28 martie 2016, un update important al aplicației Waze, face posibil ca aceasta să atenționeze șoferii atunci cât aceștia depășesc limita de viteză. Cum o face? La fel ca și celelalte date, viteza limită a segmentelor de drum este primită de la toți utilizatorii Waze. Waze nu consumă aceste date dintr-un API, ci direct din informațiile trimise de utilizatori. Cu toate acestea, acest sistem nu păstrează un istoric al încălcărilor limitei de viteză, ci doar atenționează șoferul precum că acesta depășește limita de viteză.

[14] Tehnologie folosită pentru a transmite informații despre ambuteiaje și situațiile neplăcute pe drumuri.

Figură 3.4 Exemplu de atenționare a șoferului, aplicația Waze.

Navigatoarele auto

Navigatoarele auto sunt echipamente comerciale destinate conducătorilor auto şi au ca scop ușurarea activităților specifice șoferilor. Majoritatea sistemelor de navigare, folosesc serviciile web descrise mai sus aducând îmbunătăţiri grafice, specifice în funcţie de model. Aceste sisteme oferă atât limita de viteza cât şi viteza de deplasare a autovehicului, însă nu oferă un istoric al încălcărilor. Costurile acestor produse variază în funcţie de model, caracteristici, funcționalități.

Capitolul 4. Analiză şi Fundamentare Teoretică

În acest capitol se prezintă arhitectura conceptuală a sistemului. Și mai exact, cerințele aplicației, cerințele funcționale cât și cele non-funcționale care sunt necesare. Tot aici sunt prezentate și detaliate tehnologiile folosite la dezvoltarea acestei aplicații.

4.1. Cerinţele aplicaţiei

Cerințele aplicației sunt detaliile, care se supun anumitor constrângeri în așa fel ca să ofere utilizatorului final o experiență de utilizare plăcută. Cerințele și identificarea acestora constituie o parte importantă din ciclul de proiectare a aplicației.

Orice aplicație are scopul de a rezolva anumite probleme iar modul de rezolvare a problemei/problemelor trebuie să fie cât mai eficient. Cerințele aplicației se împart în două categorii: cerințe funcționale și cerințe non-funcționale. La mod general vorbind, cerințele funcționale sunt comportamente specifice aplicației, adică funcționalitatea acesteia; în timp ce, cerințele non-funcționale, sunt alte criterii, mai puțin cuantificabile, mai puțin observabile și/sau mai puțin direct legate de scopul principal al aplicației.

Cerințele funcționale cât și cerințele non-funcționale sunt de importanță majoră în ciclul de viață a unei aplicații. Ele joacă un rol important în modul în care aplicația este răspândită la mai mulți utilizatori și utilitatea acesteia.

4.1.1. Cerinţe funcţionale ale aplicaţiei

Înregistare	Utilizatorul trebuie să aibă posibilitatea de a se înregistra, în așa fel se vor putea identifica informațiile unice, care aparțin unui anumit utilizator, inclusiv și istoricul încălcărilor.
Autentificare	Utilizatorul trebuie să aibă posibilitatea să se autentifice în sistem.
Detectarea vitezei de deplasare	Utilizatorul trebuie să ai aibă posibilitatea să afle care este viteza cu care se deplasează vehiculul.
Detectarea vitezei maxime admisibile (pentru segmentul curent de drum)	Utilizatorul trebuie să aibă posibilitatea să afle care este viteza maximă admisă cu care se poate deplasa, pe acea porțiune de drum.
Atenționarea în cazul depășirii vitezei maxime admisibile	Utilizatorul trebuie să fie atenționat atunci când acesta depășește limita de viteză permisă pe porțiunea de drum pe care acesta se află.

Salvarea locației sub formă de imagini (când s-a depășit viteza)	Atunci când Utilizatorul a depășit limita de viteză, începând cu cel puțin zece secunde, informații precum data, ora, locația sunt salvate ca mai apoi să se poată vizualiza aceste informații.
Vizualizarea încălcărilor	Utilizatorul trebuie să aibă posibilitatea să vizualizeze încălcările făcute. Încălcări ce se vor putea vizualiza sub forma de imagini, indicând data și ora când acestea au avut loc.
Primire statistici prin mail la fiecare sfârșit de săptămână (cu privire la încălcări/progrese)	Utilizatorul are posibilitatea să primească statistici, care privesc încălcările efectuate de către acesta timp de o săptămână.

Tabel 4.1 Cerințele funcționale ale aplicației

4.1.2. Cerinţe non-funcţionale ale aplicaţiei

Performanța	Performanța aplicației se referă la timpul de răspuns al acesteia și nu în ultimul rând la consumul resurselor fizice al dispozitivelor. Timpul de răspuns al aplicației trebuie să fie cât mai rapid, în așa fel experiența utilizatorului la folosirea aplicație va fi una plăcută. Totodată, întârzierile produse de aplicație, atunci când dispozitivul se află sub o rază de acoperire mai precară, și mai ales că se așteaptă primirea datelor de la server, timpul trebuie să fie cât mai redus. Evident, acest lucru nu se poate garanta în totalitate, dar datorită unor condiții impuse sistemului, acestea se pot minimiza, de exemplu prin apeluri mai rare ale serverului.
Utilizabilitatea	Această cerință non-funcțională se referă la modul în care aplicația poate fi „acceptată” de către utilizator. Aici joacă rolul și rata de adopție a aplicației care este caracteristică utilizatorilor. Cu cât aplicația are o utilizabilitate mai ușoară, cu atât rata de adopție a acesteia este mai sporită. Pentru a avea o rată de adopție sporită este strict necesar ca aplicația să fie cât mai prietenoasă. Prin prietenoasă se subînțelege: Ușor de folosit pentru orice categorie de utilizatori, de la cei mai experimentați pană la cei cu o experiență mai săracă în utilizarea aplicațiilor. Intuitivă. Utilizatorul aplicației trebuie să cunoască o experiență plăcută chiar de la prima utilizare a aplicație. Interfața joacă un rol decisiv în aprobarea succesului aplicației.

Robustețe	Este necesar ca aplicația să fie robustă. Asta înseamnă că toate erorile posibile care pot apărea sunt tratate corespunzător astfel încât experiența utilizatorului să nu fie afectată. Este nevoie ca aplicația să fie testată și nici într-un caz să afecteze funcționare dispozitivului fizic
Disponibilitate	Această cerință ne garantează că aplicația este disponibilă oricând si oriunde. Asta presupune că, oriunde nu s-ar afla conducătorul auto, acesta poate fi sigur că aplicația este utilizabilă. Acest lucru este garantat în afara cazului în care semnalul dispozitivului lipsește. Pentru utilizare este strict necesar folosirea internetului.
Eficiența	Având în vedere că aplicația este te tip client-server, iar pentru comunicarea între cele două componente este nevoie de conexiune la internet, consumul datelor care sunt necesare funcționării aplicației trebuie să fie cât mai mic.
Securitatea	Chiar dacă aplicația nu se privește ca fiind una strict confidențială, există totuși și o garantare a faptului că datele utilizatorului sunt în siguranță. Aceasta se datorează în mare parte la modalitatea de criptare a parolelor.

Tabel 4.2 Cerințe non-funcționale ale aplicației.

4.2. Cazuri de utilizare

Cazurile de utilizare oferă o descriere asupra întregului comportament al sistemului. Acestea acoperă o imagine generală privitor la cum sistemul poate fi folosit de către utilizatori. Pentru cazurile de utilizare trebuie specificați toți actorii care intră în contact cu sistemul. Cercetarea sistemului implementat a dovedit că este nevoie doar de un singur actor care va interacționa cu sistemul, care este utilizatorul simplu.

Procesul Unificat definește cazurile de utilizare, cuprinzând tot amalgamul de cazuri de utilizare. Mai mult decât atât este un model al funcționalităților sistemului și al mediului în care sistemul e utilizat. Ce este un scenariu sau o instanță de caz de utilizare? Acest scenariu este o secvență specifică de acțiuni și interacțiuni între sistem și actori. Este util să se explice și ce înseamnă un caz de utilizare. Acest termen definește colecția de scenarii fie de succes, fie de eșec.

a) Înregistreze în sistem.

Înregistrarea prevede introducerea credințealelor de către utilizator și salvarea acestora în baza de date. Utilizatorul poate face această înregistrare în mai multe moduri/combinații. Acestea pot fi: folosind email/parolă, telefon/parolă sau înregistrarea în sistem cu ajutorul rețelei de socializare Facebook. Înregistrarea este obligatorie doar o singură dată. După acest pas, utilizatorul doar va fi nevoit să se logheze în sistem.

Actor principal: Utilizator simplu

Scenariu:
Utilizatorul: dorește să se înregistreze în sistem ca să poată folosi aplicația.
Precondiții:

- Aplicația trebuie să fie instalată pe dispozitiv și rulată.
- Dispozitivul trebuie să fie configurat astfel încât să permită conexiunea aplicației la Internet, acest pas se realizează în caz că sunt unele restricții predefinite legate de securitatea dispozitivului.
- Dispozitivul mobil trebuie să fie conectat la rețeaua Internet.

Postcondiții:

- Aplicația revine la ecranul de autentificare în sistem, unde se așteaptă să se introducă credințealele cu care s-a făcut înregistrarea.
- Datele utilizatorului vor fi salvate în baza de date.

Scenariul de success:

- Utilizatorul instalează aplicația.
- Utilizatorul acordă acces la Internet și la locația dispozitivului aplicației.
- Utilizatorul rulează aplicația.
- Utilizatorul tastează butonul „Register” și își introduce credențialele unice în câmpurile cerute.
- Aplicația acceptă datele, dacă acestea sunt valide și creează o conexiune HTTP[15] la baza de date. Apoi salvează o înregistrare nouă, conținând credențialele trimise.
- Ecranul de autentificare este afișat.
- După introducerea credențialelor, aplicația se deschide.

Scenariu de eșec:

- Permisiunile de conectare la Internet și de accesare a locației dispozitivului nu sunt permise.
- Datele introduse sunt eronate (cazul în care dacă se alege combinația de înregistrare Email/parolă, și în câmpul Email nu se introduce un format Email corespunzător).
- Serverul nu este accesibil dintr-un oarecare motiv.

b) Autentificare utilizator

După ce utilizatorul s-a înregistrat cu succes în sistem, acesta se poate autentifica.

Actor principal: Utilizator simplu
Scenariu:
Utilizatorul: dorește să se autentifice în sistem.
Precondiții:

- Aplicația trebuie să fie instalată pe dispozitiv și rulată.
- Dispozitivul va fi configurat astfel încât să permită conexiunea aplicației la Internet, în caz că sunt unele restricții predefinite legate de securitatea dispozitivului.

[15] Protocol de comunicare (Hypertext Transfer Protocol)

- Utilizatorul trebuie să fie înregistrat în sistem.
- Dispozitivul mobil trebuie să fie conectat la rețeaua de Internet.

Post condiții:

- Aplicația utilizator revine la ecranul în care se afișează viteza de deplasare.

Scenariul de success:

- Utilizatorul instalează aplicația.
- Utilizatorul acordă acces la Internet și la locația dispozitivului aplicației.
- Utilizatorul rulează aplicația.
- Utilizatorul apasă butonul „Login”, unde introduce credențialele.
- Credențialele sunt acceptate și aplicația se deschide.

Scenariu de eșec:

- Conectare la Internet și accesare a locației dispozitivului nu sunt permise.
- Credințealele sunt greșite.
- Serverul nu este accesibil dintr-un oarecare motiv.

c) Preluarea informațiilor referitoare la viteza de deplasare.

După ce utilizatorul s-a înregistrat și s-a autentificat în sistem, acesta dorește să vadă care e limita de viteză cu care se deplasează.

Actor principal: Utilizator simplu.
Scenariu:
Utilizatorul: dorește să cunoască viteza de deplasare

Precondiții:

- Aplicația trebuie să fie instalată pe dispozitiv și rulată.
- Dispozitivul trebuie să fie configurat astfel încât să permită conexiunea aplicației la Internet în caz, că sunt unele restricții predefinite legate de securitatea dispozitivului.
- Utilizatorul trebuie să fie înregistrat și autentificat în sistem.
- Dispozitivul mobil trebuie să fie conectat la rețeaua de Internet.

Post condiții:

- Aplicația utilizator afișează viteza de deplasare.

Scenariul de success:

- Utilizatorul instalează aplicația.
- Utilizatorul acordă acces la Internet și la locația dispozitivului aplicației.
- Utilizatorul rulează aplicația.
- Utilizatorul se autentifică cu credențialele acestuia

- Credențialele sunt acceptate și aplicația se deschide.
- Ecranul în care este afișată viteză de deplasare apare.

Scenariu de eșec:

- Conectarea la Internet și accesare a locației nu sunt permise.
- Credențialele sunt greșite.
- Serviciu GPS nu este disponibil și viteza de deplasare nu este calculată.
- Serverul nu este accesibil dintr-un oarecare motiv.

d) Preluarea informațiilor referitoare la viteza limită de deplasare

După ce utilizatorul a aflat cu succes viteza cu care acesta se deplasează, dorește să afle și viteza maximă de deplasare permisă pe segmentul de drum pe care acesta se află.

Actor principal: Utilizator simplu.

Scenariu:

Utilizatorul: dorește să cunoască viteza maximă de deplasare permisă.

Precondiții:

- Aplicația trebuie să fie instalată pe dispozitiv și rulată.
- Dispozitivul trebuie să fie configurat astfel încât să permită conexiunea aplicației la Internet, în caz că sunt unele restricții predefinite legate de securitatea dispozitivului.
- Utilizatorul trebuie să fie înregistrat și autentificat în sistem.
- Dispozitivul mobil trebuie să fie conectat la rețeaua de Internet.

Post-condiții:

- Aplicația afișează viteza maximă de deplasare permisă.

Scenariul de succes:

- Utilizatorul instalează aplicația.
- Utilizatorul acordă acces la Internet și la locația dispozitivului aplicației.
- Utilizatorul rulează aplicația.
- Utilizatorul se loghează cu credențialele acestuia.
- Credențialele sunt acceptate și aplicația se deschide.
- Pe ecran se afișează limita de viteză.

Scenariu de eșec:

- Conectare la Internet și la accesare a locației pe dispozitiv - nu sunt permise.
- Credențialele sunt greșite.
- Nu se poate afla viteza de deplasare.
- Serviciu care oferă limita de viteză nu este disponibil.
- Serverul nu poate fi accesat dintr-un oarecare motiv.

e) Vizualizarea informațiilor referitoare la istoricul încălcărilor

Folosindu-se de ce ceva timp de aplicație, utilizatorul dorește să cunoască istoricul încălcărilor care au fost depistate în timp ce acesta conducea autoturismul.

Actor principal: Utilizator simplu.

Scenariu:

Utilizatorul: dorește să vadă un istoric al încălcărilor depistate de sistem.

Precondiții:

- Aplicația trebuie să fie instalată pe dispozitiv și rulată.
- Dispozitivul trebuie să fie configurat astfel încât să permită conexiunea aplicației la Internet, în caz că sunt unele restricții predefinite legate de securitatea dispozitivului.
- Utilizatorul trebuie să fie înregistrat și autentificat în sistem.
- Dispozitivul mobil trebuie să fie conectat la rețeaua de Internet.

Post-condiții:

- Aplicația utilizator afișează un ecran, în care se pot vedea locațiile sub formă de imagini în care sistemul a detectat limita de viteză depășită.

Scenariul de succes:

- Utilizatorul instalează aplicația.
- Utilizatorul acordă acces la Internet și la locația dispozitivului aplicației.
- Utilizatorul rulează aplicația.
- Utilizatorul se autentifică cu credențialele acestuia
- Credențialele sunt acceptate și aplicația se deschide.
- Din meniul aplicație se alege butonul “Show History”

Scenariu de eșec:

- Conectare la Internet și la accesare a locației pe dispozitiv - nu sunt permise.
- Serverul nu este disponibil și datele nu pot fi interceptate.

f) Vizualizarea informațiilor precum statistici ale încălcărilor

După o perioadă în care sistemul a fost folosit, se dorește vizualizarea statisticilor referitoare la încălcări.

Actor principal: Utilizator simplu.

Scenariu:

Utilizatorul: dorește să vadă statistici la încălcările efectuate de acesta

Precondiții:

- Aplicația trebuie să fie instalată pe dispozitiv și rulată.

- Dispozitivul trebuie să fie configurat astfel încât să permită conexiunea aplicației la Internet, în caz că sunt unele restricții predefinite legate de securitatea dispozitivului.
- Utilizatorul trebuie să fie înregistrat și autentificat în sistem.
- Dispozitivul mobil trebuie să fie conectat la rețeaua de Internet.

Post-condiții:

- Aplicația utilizator afișează statistici referitoare la încălcări.

Scenariul de succes:

- Utilizatorul instalează aplicația.
- Utilizatorul acordă acces la Internet și la locația dispozitivului aplicației.
- Utilizatorul rulează aplicația.
- Utilizatorul se autentifică cu credențialele acestuia.
- Credențialele sunt acceptate și aplicația se deschide.
- Din meniul principal al aplicației se alege “Show Statistics”.
- Apare ecranul în care sunt prezente statisticile.

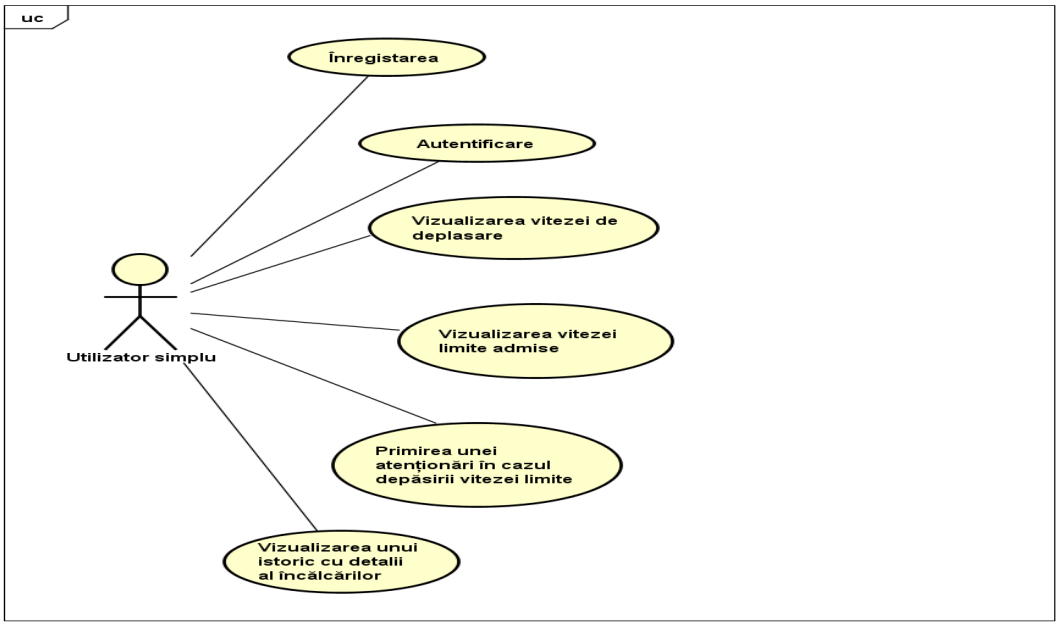
Scenariu de eșec:

- Conectare la Internet și la accesare a locației pe dispozitiv - nu sunt permise.
- Serverul nu este accesibil dintr-un oarecare motiv.

Diagrama cazurilor de utilizare

Diagrama UML [16]este o diagramă a cazurilor de utilizare care prezintă, actorii și relațiile dintre aceștia. Ca și o paranteză, vorbim de actor în limită de obiect cu un comportament specific.

[16] Unified Modeling Language – limbaj standard pentru descrierea de modele și specificații software.



Figură 4.1 Diagramă Use-Case, utilizator simplu

4.3. Fundamentare teoretică

În acest subcapitol se regăsește o sinteză teoretică a tehnologiilor incluzând și partea de design a pattern-urilor utilizate. Deci, sunt arătate tehnologiile alese și folosite la dezvoltarea sistemului la 4 nivele: structura aplicației, comunicare cu servicii Web, interfața și nu în ultimul rând legăturile dintre acestea.

Tehnologii şi resurse uzilizate

În această secție, se vor prezenta tehnologiile și deciziile de design realizate pentru aplicație. Urmează să fie atinse și subiecte legate de aplicațiile native versus aplicațiile hibride. Mai mult decât atât, se vor prezenta atât framework-ul cât și tool-urile utilizate la implementarea noii aplicații.

Cordova

Apache Cordova, conform sursei [10] și sursei [14], este un framework de dezvoltare a aplicațiilor mobile de tip hibrid care permite dezvoltatorului să acceseze funcționalitățile dispozitivelor native cum ar fi: camera sau accelerometru direct în JavaScript. Aplicațiile mobile, care folosesc acest framework sunt construite cu tehnologiile web cum ar fi HTML, CSS și JavaScript. Cordova asigură dezvoltarea aplicației fără ca să aibă nevoie de a face cu codurile native cum ar fi Java, Objective C, [17]și permite dezvolatarea doar numai cu utilizarea tehnologiilor web. Cordova utilizează un SDK pentru a introduce dezvoltarea aplicației într-un container nativ. Aceasta face ca aplicația să poată să fie distribuită prin magazinul de aplicații și să utilizeze caracteristicile native. Corodva are un suport stabil de la platformele precum iOS,

[17] Limbaj de programare Apple.

Android, Blackberry, Windows și altele. Toate caracteristicile native sunt accesate prin utilizarea plugin[18]-urilor, mai mult ca atât Cordova are deja câteva plugin-uri de bază. Există mai mult de 900 de plugin-uri Cordova, iar acestea sunt construite în codul nativ și sunt dependente de platformă. Dacă dorim să găsim un plugin de care avem nevoie, trebuie să verificăm dacă platforma aleasă are plugin-uri create pentru Cordova în acel moment. De exemplu, un plugin Cordova existent este *cordova-googlemaps-plugin.*

Figură 4.2 Exemplu de arhitectură a unui plugin Cordova

Sursă imagine - [14]

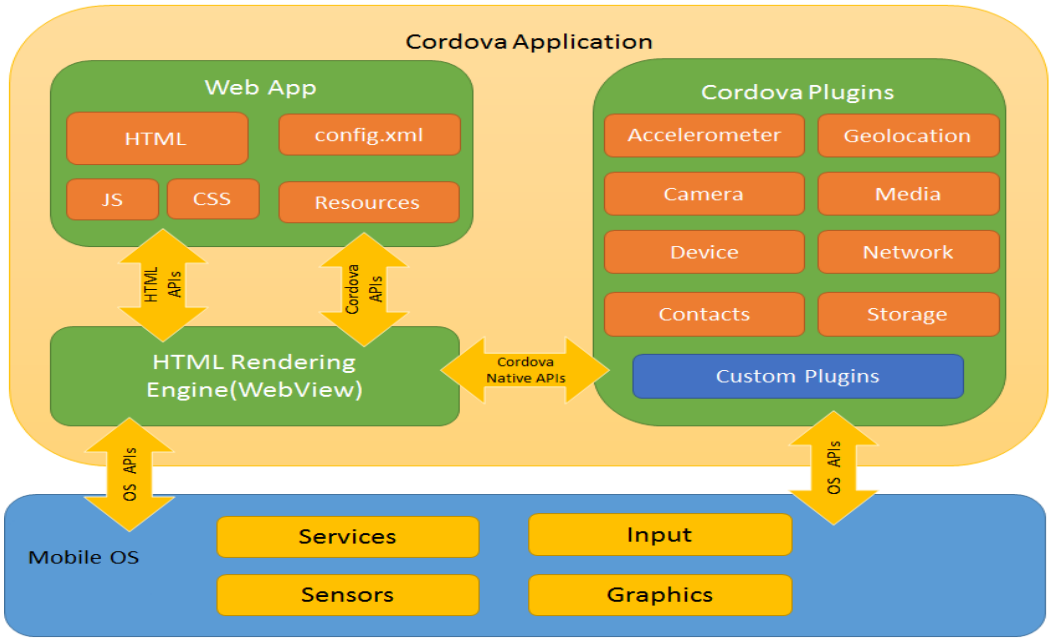
În tabelul de mai jos se pot vedea care sunt caracteristicile native ale dispozitivului care pot fi accesate din JavaScript.

Caracteristică	Desciere
Accelerometru	Permite acces la locația, mișcările și orientarea dispozitivului
Camera	Permite utilizatorului să facă imagini și să le acceseze din dispozitiv
Compasul	Returnează direcțiile compasului din dispozitiv
Conexiunea	Permite aplicației să determine dacă există conexiune la Internet și cât de bună este aceasta
Date dispozitiv	Returnează metadate despre dispozitiv
Fișiere	Permite accesul la citirea și scrierea fișierelor și directoarelor
Evenimente	Suport pentru o varietate de evenimente, precum datele despre procentajul bateriei
Geolocație	Permite localizarea dispozitivului
InAppBrowser	Permite acces la folosirea unui navigator web direct din aplicație
Media	Oferă suport pentru înregistrările audio.
Notificări	Suport pentru metodele de a notifica

[18] Integrarea unor funcționalități în alte programe.

	utilizatorul, incluzând sunetele și vibrația
Stocare	Permite crearea unei baze de date locale pentru aplicație

Tabel 4.3

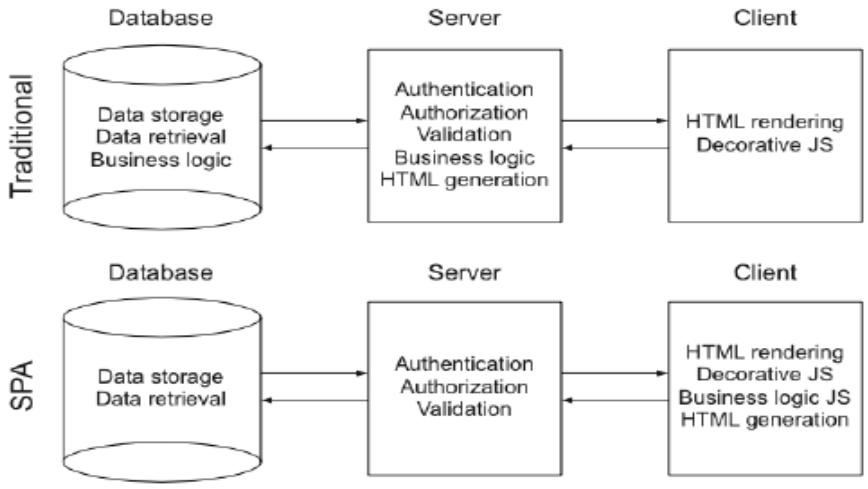


Figură 4.3 Arhitectura Cordova

Sursă imagine - [14]

Cordova Single Page Application

Odată ce Cordova utilizează tehnologiile web pentru a construi o aplicație, vom utiliza abordarea numită SPA (Single Page Application), pentru crearea aplicației. Conform SPA de Mikowski și Powell (sursa [10]), SPA este o modalitate de livrare a conținutului navigatorului web, care nu trebuie să se reîncarce în timpul când pagina este utilizată. Prin reîncărcare se are în vedere că pagina nu este reîncărcată la interacțiunea cu utilizatorul. Această abordare sau situație oferă utilizatorului o senzație că ar utiliza o aplicație nativă. De asemenea se evită redarea conținutului de pe partea de server, conținutul va fi redat pe partea clientului. Datorită acestei abordări, reducem gradul de informație transferată între client și server și lăsăm ca serverul să se focuseze pe trimiterea informațiilor relevante decât pe butoane sau alte componente care ar trebui să intercepteze. Figura de mai jos ilustrează cum aplicația SPA funcționează în teorie și cum logica de business și redarea HTML-ului este transferată pe partea clientului și cum serverul este focusat/concentrat pe trimiterea și primirea informației.



Figură 4.4 Aplicații web tradiționale vs aplicații SPA

Interfața grafică. Ionic

Utilizarea tehnologiilor web poate ușura mult activitățile necesare la implementarea unei aplicații. Spre exemplu, un avantaj este existența multor framework-uri UI disponibile, pentru implementarea aplicațiilor. Un foarte cunoscut și apreciat framework este Bootstrap[19], care furnizează predesign-ul componentelor cum ar fi – butoanele, titlurile, tabelele și alte aspecte de acest gen. Aceste componente pot fi ușor customizate/îmbunătățite cu ajutorul CSS-ului. Conform sursei [6], Bootstrap este creat cu un principiu mobil unde componentele sunt optimizate special pentru telefoanele mobile. Totuși, framework-ul este format pentru a crea aplicații web care sunt accesate și de pe un browser web și nu doar de pe telefoane inteligente.

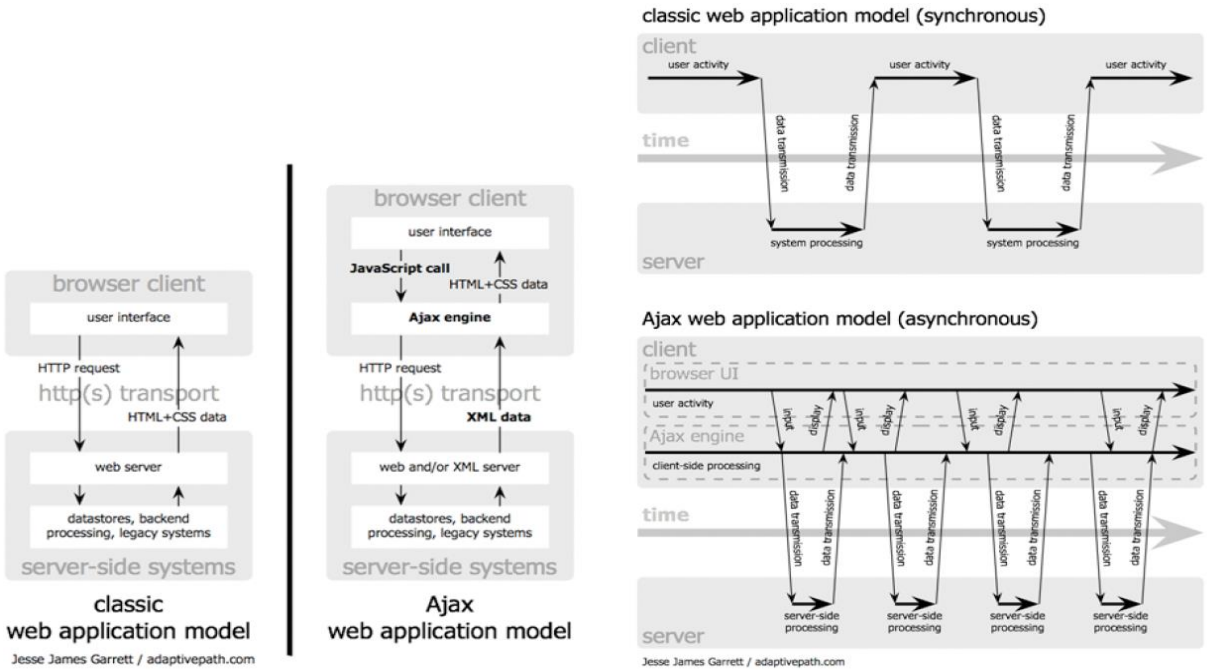
S-a dorit un UI[20] care să ofere interfață utilizatorului cât se poate de nativă, și atât butoanele cât și meniul să aibă tangență cu aplicațiile native. Conform sursei [13], Framework-ul Ionic este un framework hibrid de aplicație mobilă care este focusat pe construcția aplicațiilor mobile. Scopul acestuia este de a oferi senzația de aplicații native și anume, prin deținerea unor componente care arată la fel ca componentele native. Motivul pentru care s-a ales Ionic pentru construirea aplicației este că majoritatea dintre utilizatori au telefoane mobile cu iOS și prin urmare Ionic oferă componente UI care arată ca și componentele native reale.

JavaScript

Cu HTML și CSS se poate prezenta interfața aplicației, pe când JavaScript susține interacțiunea dintre aplicație și utilizator. JavaScript la fel se ocupă și de comunicarea dintre client și server utilizând modelul de programare de tip Asynchronous JavaScript[21] și XML[22] (AJAX[23]), la fel cum se descrie și în sursa [18]. Pe o aplicație web tradițională, când utilizatorul apasă pe un buton, clientul (browser-ul web) trimite o cerere serverului

[19] Framework UI dezvoltat de Twitter
[20] User Interface sau interfața grafică a utilizatorului.
[21] Comunicare asincronă între componente.
[22] Extensible Markup Language – meta-limbaj de marcare
[23] Asynchronous JavaScript and XML

(serverul web) și atunci utilizatorul este în așteptarea unui răspuns. Când serverul este pregătit să ofere un răspuns, clientul primește câteva date trimise de către acesta. Prin utilizarea modelului de comunicare AJAX, utilizatorul are sentimentul că pagina nu așteaptă pentru răspuns sau nu are nevoie să fie redirecționată și prin urmare face ca aplicația să fie nativă când este accesată de pe un dispozitiv mobil. O ilustrare a modelului de programare AJAX și cum AJAX interacționează la comunicarea dintre client și server este prezentat în figura de mai jos.



(a) Modelul AJAX (b) Comunicare Sincronă vs Comunicare Asincronă

Figură 4.5 Modelul de programare AJAX
Sursă http://www.billnsara.com/js/ajax.asp

Angular JS

Framework-ul pentru UI menționat anterior, a fost construit cu Angular JS [24] și depinde în totalitate de funcționalitățile Angular JS. Conform resursei [23], Angular JS este un framework de tip open source și menținut de către Google. Câteva din caracteristicile AngularJS:

- **Legătură bidirecțională** (two-way binding) - oferă sincronizare automată între paginile HTML și model (Data).
- **Structura pentru codul de front-end** - prin divizarea codului în servicii, controlere și pagini web.
- **Suportă rutarea** – putem defini rutele proprii și face ca aplicația să se simtă nativă și deci, nu avem nevoie să depindem de un server web care ar furniza resursele corecte.
- **Șablon HTML**– putem scrie expresii în file-urile HTML și atunci paginile sunt randate pe condițiile definite.

[24] Framework JavaScript dezvoltat de Google.

- **Validare formelor** – spre exemplu am putea cere un număr prezentat într-o formă și apoi Angular poate valida numărul scris în acea formă.
- **Directive HTML** – putem în AngularJS să definim sintaxa sau directive HTML.
- **Dependency Injection** – una dintre cele mai bune caracteristici ale Angular care oferă dependency injection pe toate serviciile și controalele.

Există și alte framework-uri similare cu AngularJS cum ar fi EmberJS și Backbone, dar având în vedere că Ionic are nevoie de funcționalitățile AngularJS nu am luat în considerare și alte framework-uri JavaScript și prin urmare am utilizat AngularJS ca și framework de bază JavaScript pentru aplicație.

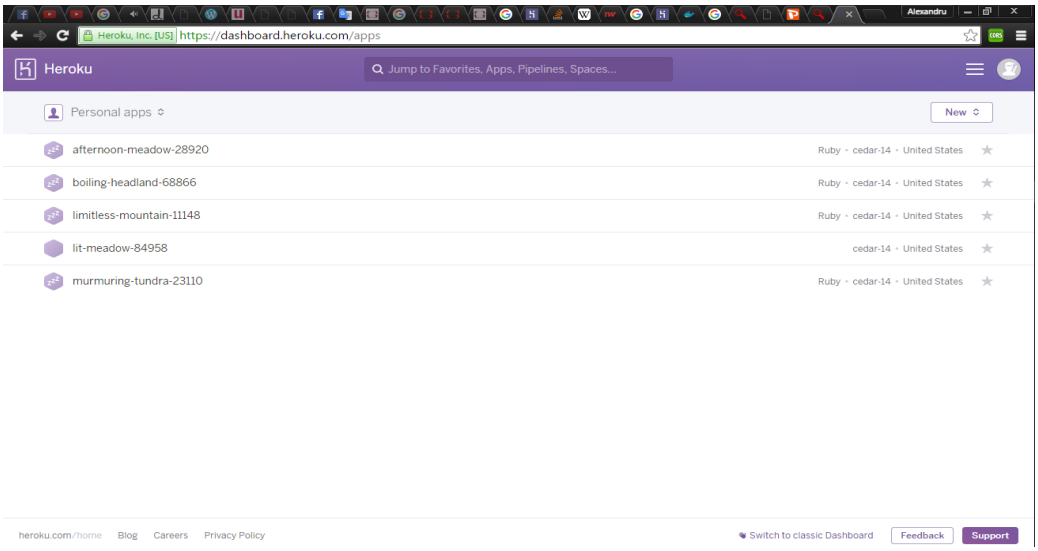
Heroku

Potrivit lui Orion Henry, unul dintre cei trei co-fondatori, Heroku este un cuvânt format din „hero” și “haiku”, motivul fiind datorat unui inventator al limbajului de programare Ruby. Fondată în 2007, serviciul cu numele Heroku, conform resursei [21], este defapt un PaaS (Platform as a Service care te ajută să-ți rulezi aplicația proprie într-un Cloud[25]). În alte cuvinte, Heroku face posibil ca aplicația dezvoltată să-și găsească loc în Internet, fără a interacționa pe partea de server. Datorită aplicației oferite de Heroku, în doar câteva momente poți face ca aceasta să fie accesibilă în orice zonă de pe glob, ca și cum s-ar accesa o pagină web. Desigur, există și alternative pentru Heroku, cum ar fi Appfog sau Dotcloud, care sunt servicii asemănătoare însă sunt mult mai tinere și nu atât de populare. Evident, sunt alternative la fel de bune însă mai costisitoare, cum ar fi Amazon Web Services ori servicii de host precum DreamHost. Totuși, acestea sunt mult mai complicat de administrat și după cum s-a scris, mult mai scumpe. Heroku poate fi folosit de către orice dezvoltator, sau orice doritor care nu îi convine să administreze propriul său host, întrucât acesta necesită atât costuri suplimentare cât și cunoștințe pentru a fi administrat. Problema de administrare devine și mai dificilă atunci când numărul de utilizatori care folosesc serviciu hostat local crește. Odată cu creșterea acestora, problemă devine critică. Apare necesitatea de a implementa termeni precum “Load balancing[26]”, iar procesul de implementare nu este unul ușor. Heroku, ca și celelalte cloud-uri actuale, se bazează pe tehnologii precum „Containerizarea”, care oferă posibilitatea de a migra containerele în interiorul unui Cloud, în așa mod oferindu-se posibilitatea de a avea „Load balancing” iar serverul alocă automat resursele necesare pentru această cerință.

Cum funcționează Heroku?

Când se creează o aplicație pe Heroku, acesta publică aplicația în Stiva Cedar (Cedar Stack), care este un mediu online de rulare a aplicațiilor și suportă aplicații dezvoltate în: Java, Node.js, Scala, Python, RubyOnRails, PHP. Odată ce aplicația a fost creată, Heroku asignează un nume unic acestei aplicații. În Figura 4.6 se poate vedea

[25] Computerizare în nori. Concept modern în domeniul computerelor și informaticii.
[26] Soluție pentru suprasolicitarea unui serviciu



Figură 4.6 Exemplu de aplicații publicate pe Heroku

Când vorbim de aplicații care rulează în Heroku, acestea sunt împărțite pe o mulțime de servere virtuale, sau alt fel numite – instanțe, care pot fi pornite sau oprite în orice moment. Aceste instanțe în contextul Heroku sunt numite "dynos" care sunt acele containere de care s-a vorbit mai sus. Aceste containere, pot fi dotate cu opțiuni de monitorizare a aplicației, cum ar fi: câte ore pe zi să funcționeze aplicația sau când să ruleze aplicația (la un timp predefinit).

Heroku și Git

Unul dintre motivele pe care-l fac pe Heroku să fie atât de popular și de folosit, este folosirea versiunii de control, care este o modalitate de administrare a codului, așa cum este Git[27]. Aplicațiile pot fi publicate numai dacă se folosește Git pentru administrarea codului. Odată ce avem realizată această restricție, cu o singură comandă se poate publica sistemul în Cloud.

Cloudinary

Este o soluție pentru managementul imaginilor pentru site-uri sau aplicații mobile. Acest serviciu cuprinde toate operațiile asupra imaginilor, cum ar fi: încărcarea imaginilor, salvarea imaginilor, manipularea și optimizare acestora. Folosind acest serviciu, se poate foarte ușor încărca imagini în Cloud și manipularea acestora în mod automat, fără a folosi un software aparte. Conform resursei [20], Cloudinary oferă niște servicii web foarte puternice și care se pot integra într-o mulțime de tehnologii, cum ar fi: Ruby On Rails, jQuery, PHP, Node.js, Java, django, .NET, Android, iOs, Angular JS, Scala, Magento, WordPress. Cu alte cuvinte, dacă e nevoie ca proiectul să folosească careva imagini, în mod sigur se poate alege Cloudinary. Această alegere evidențiază prin reducerea efortului de a administra imaginile.

[27] Sistem de control al sursei

Cloud9 IDE

Conform resursei [22], acest serviciu este un IDE online, care suportă sute de Conform resursei [22], acest serviciu este un IDE online, care suportă sute de limbaje de programare, cum ar fi: PHP, Ruby, Perl, Python, JavaScript, Node.js, Go. Acest IDE oferă posibilitatea de a se începe a scrie cod direct după ce mediul de programare a fost creat. Ca să fie mai clar, acest utilitar nu trebuie configurat pentru o anumită tehnologie. El vine pre-configurat cu toate compilatoarele și utilitățile necesare. Pentru managementul mediilor de programare, acesta folosește containere Docker[28].

PHP

Conform caracteristicilor prezentate în sursa [3] și [17], PHP este un limbaj de scriptare pe partea de server. Cu PHP se poate face o mulţime de lurcuri, cum ar fi: operaţii pe imagini, salvarea fişierelor pe server, operaţii pe baza de date, managementul de email, generare de PDF, logare securizată, parsare de RSS (familie de formate de fluxuri web, realizate în format XML), implementarea de servicii SOAP şi altele. Aceste operaţii nu pot fi autorizate ca să fie implementate pe partea de client, în sistemul realizat fiind aplicaţia utilizatorului, realizată cu tehnologii JavaScript. Câteva dintre motivele care au condus la implementarea serverului folosind acest limbaj, sunt următoarele:

Scalabilitatea.

Este foarte simplu de scalat o implementare PHP.

Ușor de programat.

PHP este foarte simplu de utilizat, de aceea cei mai mulți programatori lucrează pe PHP.

Proiectat pentru Internet.

PHP este proiectat din start pentru a funcţiona pe Internet. Acesta nu este precum JAVA, care este peste tot. Funcționalitățile acestuia pot fi folosite doar pe Internet, iar asta face ca PHP să devină tot mai performant în acest domeniu.

Dezvoltare rapidă.

Folosind PHP, implementarea unei soluţii este mult mai rapidă decât folosind alte limbaje.

Manipulare ușoară a bazei de date.

Operațiile asupra bazei de date se fac într-un mod cât mai simplu.

Suport pentru programarea orientată pe obiecte.

La început PHP nu a fost un limbaj orientat pe obiecte. Acest factor a dus la pierderea din popularitate a PHP-ului atunci când POO[29] a devenit popular pe o scară foarte largă în rândurile dezvoltatorilor. Începând cu versiunea 5.0, PHP a introdus suport pentru această arhitectură iar asta l-a făcut să crească din nou în popularitate.

Framework-urile

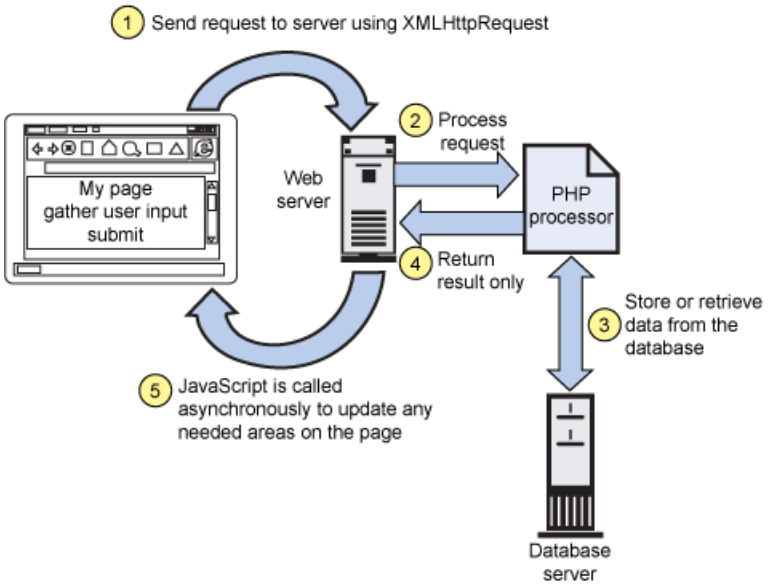
Sunt o mulţime de framework-uri pentru PHP. Acestea sunt de mare ajutor dezvoltatorilor, întrucât pot reduce considerabil timpul de dezvoltare.

AJAX şi PHP

[28] Platformă de containerizare

[29] Programare orientată obiect

Apelurile AJAX sunt foarte ușor procesare în PHP (sursa [19]). Sistemul implementat a avut nevoie de așa ceva întrucât majoritatea apelurilor folosesc AJAX.



Figură 4.7 Apel AJAX procesat în PHP
Sursă: http://www.ibm.com/developerworks/library/wa-aj-php/

SendGrid

SendGrid oferă un serviciu de livrare pe e-mail bazat pe Cloud. Acest serviciu gestionează diverse tipuri de e-mail, inclusiv notificările de livrare, cereri de prietenii, confirmări de înscrieri și email-uri publicitare. Serviciu oferă și posibilitatea de a vedea statistici cu numărul de email-uri care au fost deschise, numărul de dezabonării, raporturi spam. Acest serviciu a fost integrat în sistem pentru că una din cerințele funcționale ale acestuia este trimiterea de mail la fiecare sfârșit de săptămână către utilizatori, care să conțină statistici ale încălcării limitei de viteză.

Concluzie

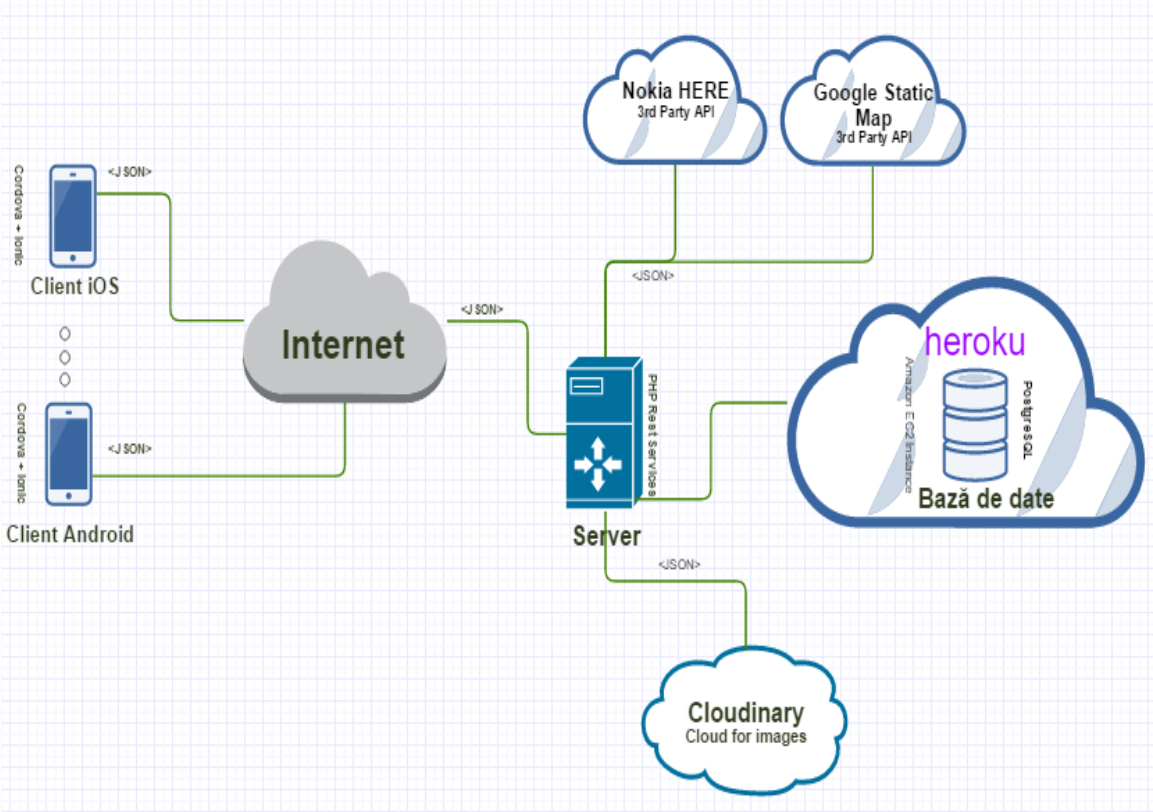
Aplicația a fost construită cu teoria, modelele de design, tehnologiile și framework-urile mai sus descrise împreună. Datorită acestor tehnologii s-a construit o aplicație mobilă de tip hibrid. Am utilizat AngularJS ca și framework-ul de JavaScript. Ionic ca și framework-ul care servește la interfața grafică, construit din elemente HTML și CSS pentru a furniza o interfață grafică cât mai aproape de interfața grafică nativă. Am utilizat Single-Page-Application și modelul de programare AJAX pentru comunicarea cu serverul web, pentru ca aplicația reală să fie asemenea unei aplicații native.

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol se descrie și se explică diagramele reprezentative pentru sistemul de supraveghere. Pe lângă acestea, se va explica și majoritatea deciziilor importante luate în procesul de proiectare, precum și descrierea componentelor, a claselor și metodelor care au avut un rol esențial în implementare.

5.1. Arhitectura conceptuală a sistemului

Arhitectura aplicației este de tip Client-Server. Aceasta presupune că aplicația se supune conceptului de sistem distribuit în care partea de client și partea de server sunt văzute ca și componente atât logice cât și fizice diferite. În cazul aplicației date, serverul este implementat și rulat pe o platformă Cloud (PaaS)[30], care la momentul de față este una din cele mai populare platforme Cloud, cu peste 50 000 de aplicații publicate. Serviciul REST-API[31] ce este implementat pe partea de server, este construit cu ajutorul tehnologiei PHP. Arhitectura conceptuală a sistemului se poate vedea în Figura 5.1.



Figură 5.1 Arhitectura conceptuală a sistemului

[30] Platformă Cloud - Platform as a Service
[31] Representational state transfer

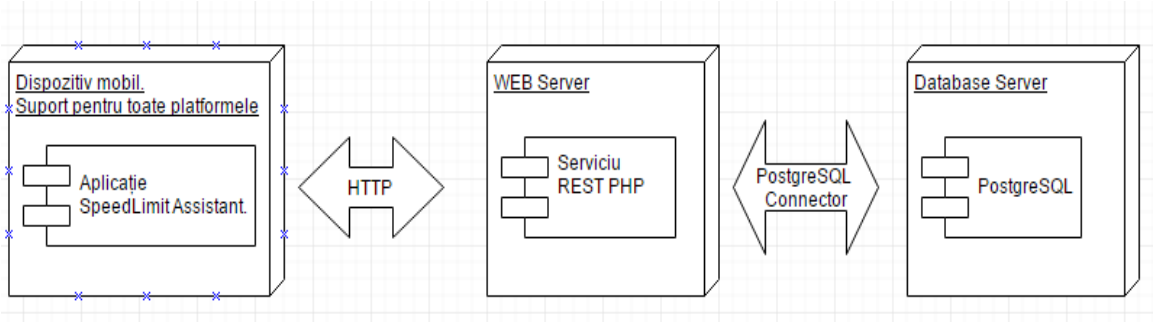
5.1.1. Rolurile componentelor care fac parte din sistem

Pentru o mai clară viziune despre fiecare componentă din sistem, este necesar de a se prezenta imaginea completă în legătură cu distribuirea celor mai importante funcționalități între componente.

Decuplarea cât mai evidentă între componentele sistemului și executare a cât mai multor operații pe partea de server aduc un număr considerabil de avantaje. Acestea ar fi:
-ușurința la modificarea aplicației server comparativ cu schimbarea fiecărui client în parte,
- odată ce dispozitivele mobile au o putere de procesare mai mică vor reuși într-un timp mai scurt să își atingă limitele.

5.1.2. Diagrama de deployment

O diagramă de deployment reprezintă în limbajul Unified Modeling (UML) - componentele hardware care apar în proiect împreună cu componentele software care rulează pe fiecare nod reprezentat de o componentă hardware și modul în care nodurile sunt conectate.



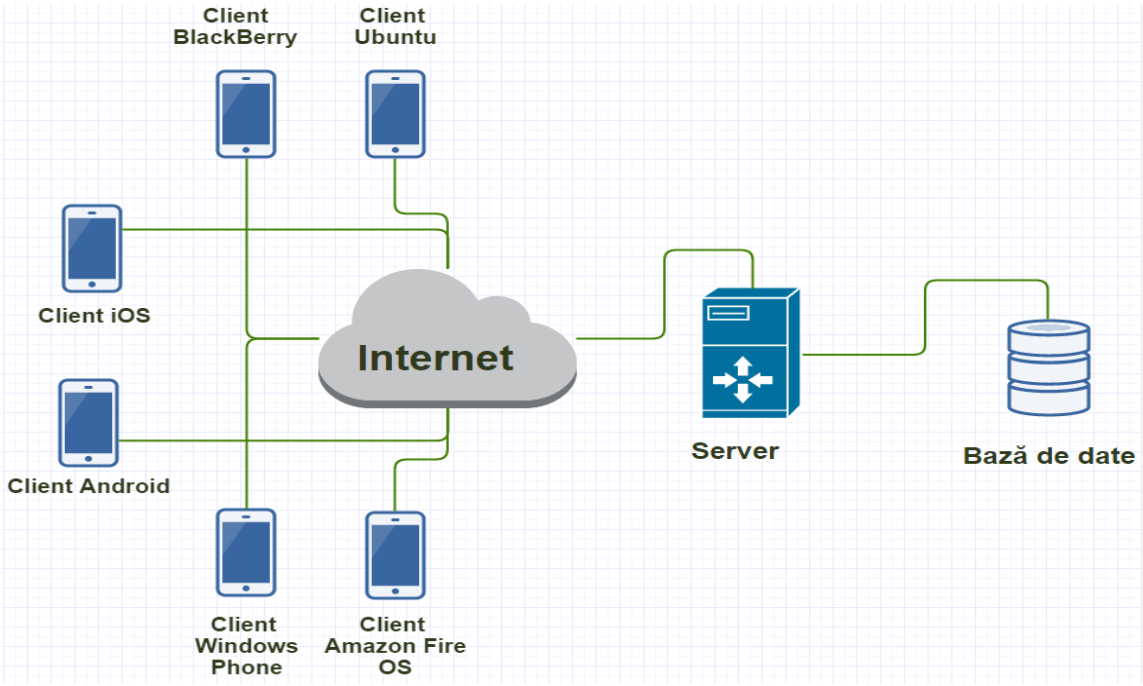
Figură 5.2 Diagrama de deployment a sistemului

5.2. Aplicaţia Client

Partea de client este implementată cu ajutorul framework-urilor Cordova + Ionic care împreună oferă o posibilitate de dezvoltare a unor aplicații mobile hibride dar care se sunt cât mai aproape de cele native, folosind tehnologiile HTML, CSS, Javascript. Datorită acestor tehnologii, aplicația client, odată implementată și testată poate fi instalată pe toate platformele mobile cunoscute în prezent, acestea fiind: Amazon Fire OS, Android, BlackBerry, Firefox OS, iOS, Ubuntu, Windows Phone, Tizen. De aici rezultă că nu este nevoie de resurse obligatorii cum ar fi: timp, cunoștințe necesare legate de o anumită platformă sau tehnologie, resurse fizice pentru a dezvolta o aplicație pe platforme diferite, cât și echipamente fizice caracteristice pentru o anumită platformă.

Comunicarea între aceste două părți, partea de client și partea de server se face prin metodele HTTP: Get, Post, Put, Delete. Aceste metode sunt îndeajuns pentru implementarea unui serviciu REST funcțional. În Figura 5.2 este ilustrată arhitectura client-server specifică aplicației. Pe partea de client avem platformele mobile pe care va

fi instalată aplicația. Pe partea de server avem serviciul REST, implementat în PHP. Mesajele trimise, cu care comunică aceste componente sunt mesaje în format JSON.



Figură 5.3 Arhitectura Client-Server specifică aplicației.

S-a încercat o realizare a cât mai multe operații pe partea de server decât pe partea de client. S-a optat pentru această realizare din cauză că, consumul de date al dispozitivului clientului trebuie să fie cât mai redus. Din această cauză, operațiile precum: realizarea imaginilor pe baza locației, trimitere mail-uri – se fac de către partea de server. Mai jos sunt descrise detaliile operațiilor pe care aplicația client trebuie să le realizeze.

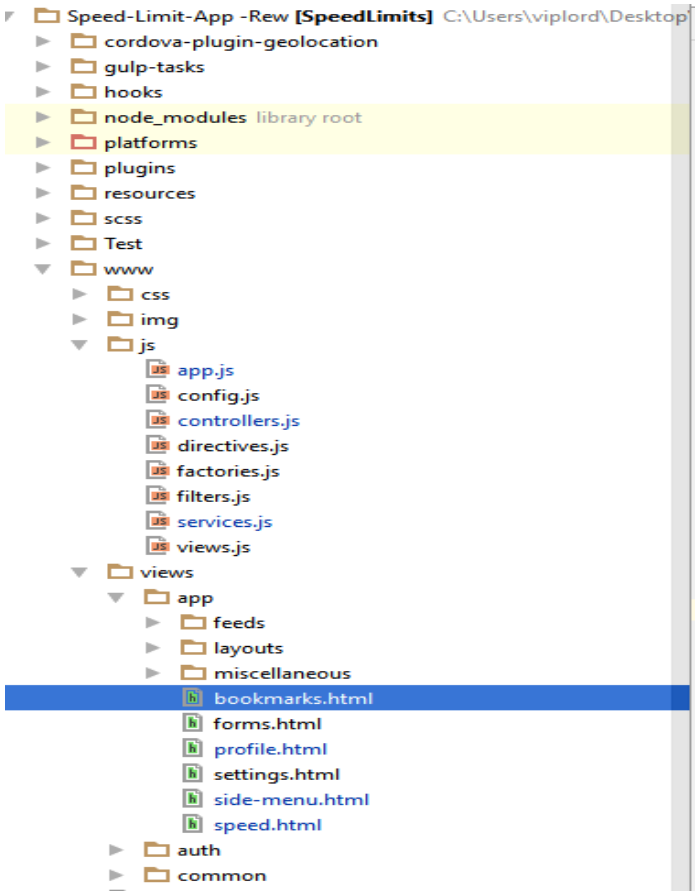
- Trimiterea de informații către server cu privire la acțiunile utilizatorilor, cum ar fi, informații legate de înregistrare sau autentificare, informații despre încălcări a limitei de viteză.
- Aplicația client afișează informațiile provenite de la server. Acestea sunt informații ce țin date despre limita de viteză, istoric și altele.

Detalii de implementare

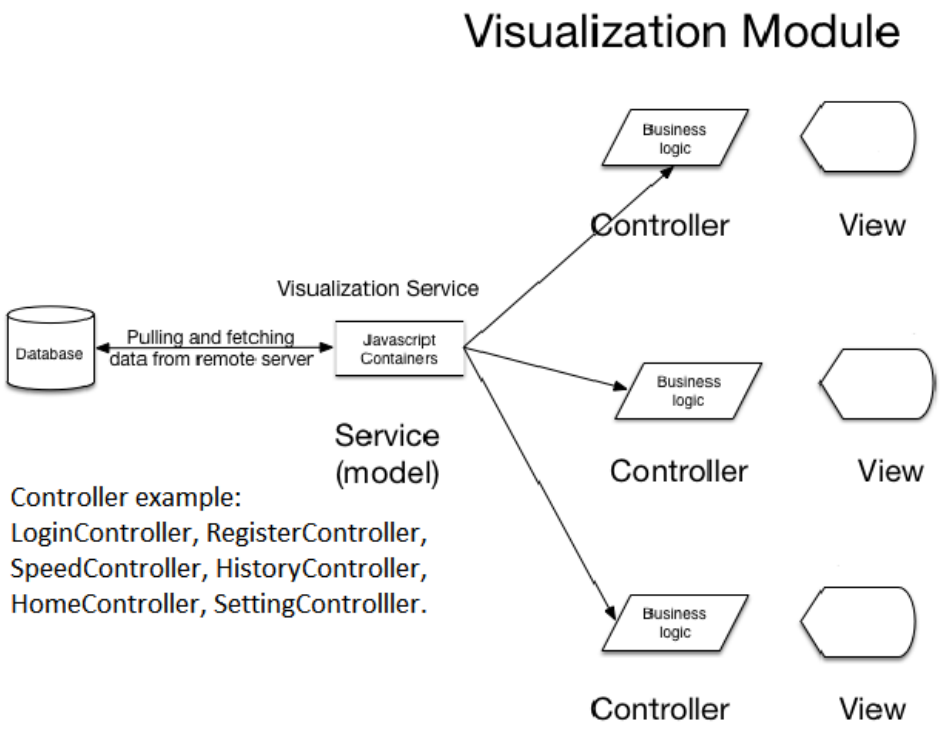
Cu ajutorul mai multor framework-uri menționate anterior, și existența plugin-urilor oferite de Cordova, s-a utilizat toate tehnologiile prin combinație și s-a reușit să se construiescă această aplicație. Cu Cordova am publicat aplicația într-un container nativ. Framework-ul JavaScript de bază a fost AngularJS. Apoi am organizat structura de fișiere ale aplicației în foldere, pe module de bază. Implementarea corespunde cu modelul de arhitectură MVC[32], unde controlerele sunt de tip JavaScript din contextul AngularJS, iar interfețele grafice sunt pagini HTML, redate cu ajutorul Ionic. Am realizat module de

[32] Model View Controller – design pattern

bază pentru aplicația mobilă prin divizarea fiecărei caracteristici în module și directoare separate. În figura de mai jos se poate vedea structura de fișiere obținută.



Figură 5.4 Structura arborelui de fișiere



Figură 5.5 Structura MVC a aplicației

Stări și rute

De obicei, la construirea aplicațiilor web, conținutul este prezentat pe baza a ceea ce utilizatorul cere de la server. Totuși, când aplicația este utilizată în interiorul containerului Cordova pe un telefon mobil, resursele sunt accesate ca și un file de sistem. Aici nu se găsește o bară de adresă unde ai putea introduce o adresă către o resursă pe care se dorește a fi accesată. Toate resursele și fișierele sunt rezolvate de către containerul Cordova utilizând navigatorul web a telefonului mobil. Pagina de start a unei aplicații hibrid, în Cordova este index.html, până în momentul când specificăm altceva. Dacă se dorește navigarea către un alt fișier, va fi necesară crearea unui buton care va accesa o adresă URL ce ar permite accesul la alte file-uri, și astfel se va rezolva cererea. Containerul Cordova lucrează ca și un server web, servind conținutul serverului bazat pe resursele cerute de către utilizator. Se dorește ca aplicația să fie cât mai nativă iar abordarea de mai sus nu va oferi utilizatorului această părere. Angular a construit un modul pentru rutare, numit ngRoute. Acest modul permite afișarea conținutului bazat pe ce resursă vrem să vedem și la fel care interfață este prezentată.

Când se construiesc aplicații largi, URL-urile folosite pot deveni lungi și complicat de menținut. În special când se trimit cereri lungi prin URL. În loc de a lucra cu URL-uri, Angular UI permite să se lucreze cu stări. Este nevoie numai de a se specifica varietatea stărilor și a resurselor, si nu în ultimul rând librăria care se ocupă de cererile resursei. Un exemplu elocvent ar fi când se utilizează AngularUI routes, pentru definirea varietăților de rute. În figura de mai jos se observă un exemplu de 3 stări diferite și cum URL-ul este încă neschimbat.

```
.config( function($stateProvider, $urlRouterProvider,
$ionicConfigProvider, $httpProvider) {
    $httpProvider.defaults.useXDomain = true;
    delete $httpProvider.defaults.headers.common['X-Requested-With'];
    $stateProvider
      .state('auth', {
        url: "/auth",
        templateUrl: "views/auth/auth.html",
        abstract: true,
        controller: 'AuthCtrl'
      })
      .state('auth.login', {
        url: '/login',
        templateUrl: "views/auth/login.html",
        controller: 'LoginCtrl'
      })
      .state('auth.signup', {
        url: '/signup',
        templateUrl: "views/auth/signup.html",
        controller: 'SignupCtrl'
      })
        .state('app.speed', {
          url: "/speed",
          views: {
            'menuContent': {
              templateUrl: "views/app/speed.html",
              controller: 'SpeedCtrl'
        } } })
```

Exemplu de rutare în AnguladJS

```
.controller('SignupCtrl', function ($scope, $state) {
    $scope.user = {};
    $scope.user.email = "";
    $scope.user.password = "";
    $scope.doSignUp = function () {
        var str_json = JSON.stringify($scope.user)
        // JQuery loads data from server
        $.ajax({
           url:  'https://phprestapi-viplord.' +
                 'c9users.io/register_script.php',
           data: 'user=' + str_json,
           dataType: 'json', // Choosing a JSON datatype
        })
          // Data contain response from server side.
        .done(function (data) {
               if (data == 'Succes!') {
                    $state.go('app.speed');
                }
                else {
                    alert("Something wrong")
               }
         });  } })
```

Exemplu de cod JavaScript specific controlerului pentru înregistrarea utilizatorilor în sistem

```
<ion-view class="signup-view auth-view" cache-view="false">
  <ion-content scroll="false">
    <div class="row">
      <div class="col col-center">
        <div class="card sign-up-container">
          <form name="signup_form" class="" novalidate>
            <div class="item item-body">
              <label class="item item-input">
                <input type="email" placeholder="Email"
                       name="user_email"
                       ng-model="user.email" required>
              </label>
              <label class="item item-input" show-hide-container>
                <input type="password" placeholder="Password"
                     name="user_password" ng-model="user.password"
                     required show-hide-input>
              </label>
            </div>
            <div class="item item-body bottom-content">
              <button type="submit"
                    class="button button-assertive button-block"
                      ng-click="doSignUp()"
                     ng-disabled="signup_form.$invalid">
                Sign Up
              </button>
            </div>
          </form>
        </div>
        <div class="alternative-actions">
          <a class="log-in button button-small button-button-light"
           ui-sref="auth.login">
            Log In
          </a>
        </div>
      </div>
    </div>
  </ion-content>
</ion-view>
```

Cod specific paginilor HTML, în contextul Ionic

Majoritatea funcționalităților aplicației au necesitatea de conexiune la Internet pentru o bună și eficientă funcționare. Mai mult ca atât, cererile la server se realizează asincron, prin utilizarea tehnologiei AJAX.

Pentru crearea unei cereri la server, componenta este obligată să realizeze un obiect de cerere și să seteze anumite câmpuri, precum – numele metodei apelate de pe server cât și parametrii acesteia. Al doi-lea pas este serializarea[33] obiectului în format JSON și expedierea cererii la server. Ca și consecință, serverul generează un răspuns JSON care conține un mesaj cu informații utile. Răspunsul se deserializează[34] sub forma unui obiect, verificând totodată dacă conține o eroare. În cazul depistării unei erori, se

[33] Transformarea unui obiect într-o secvență de octeți

[34] Proces invers al serializării. Reface starea originală a obiectului.

creează o excepție specifică în acord cu mesajul de eroare și apoi se va afișa utilizatorului. O altă situație ar fi atunci când răspunsul nu are nici o eroare, prin urmare el se deserializează într-un obiect de model și apoi se va returna obiectul serviciului respectiv.

```
$scope.user.username  = "Alex123";
$scope.user.password  = "AlexPass";
var str_json = JSON.stringify($scope.user)

    $.ajax({ url:'https://ilies.me/register_script.php',
         data: 'user=' + str_json,
         dataType: 'JSON',
    })
    .done(function (data) {
           if (data == 'Succes!') {
               $state.go('app.speed');
            }
            else {
               alert("Email already exists")
            }
     });
```

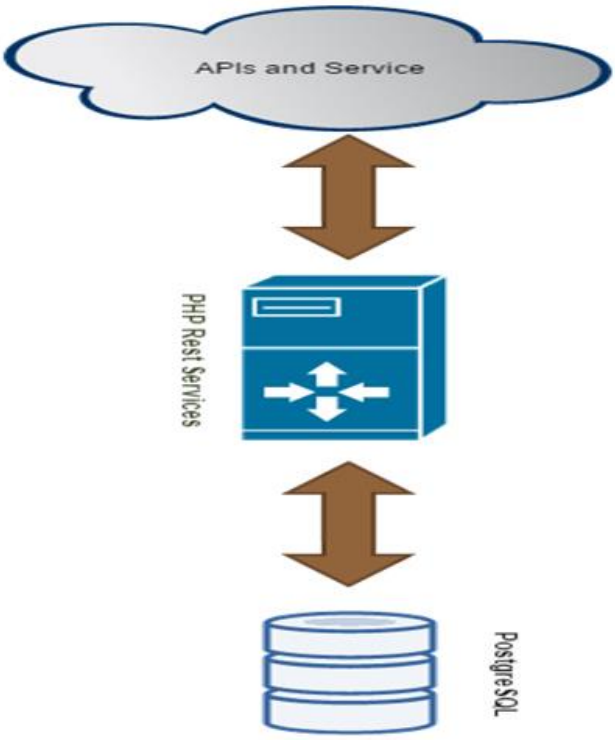
Exemplu de apel AJAX

5.3. Aplicaţia server

Arhitectura aplicaţie server

Dintre operațiile care sunt efectuate pe parte de server, sunt:

- Managementul utilizatorilor
- Consumarea de 3rd-Party APIs
- Procesarea datelor provenite de la client. În caz ca s-a detectat depășirea vitezei maxime permisă, serverul creează imagini cu locația folosind coordonatele geografice apoi le încarcă pe acestea într-un Cloud de imagini.
- Serverul creează statistici conținute din date precum încălcările limitei de viteze.
- Serverul e programat să trimită e-mail către fiecare utilizator la sfârșit de săptămână cu statistici.



Figură 5.6 Arhitectura server a aplicației

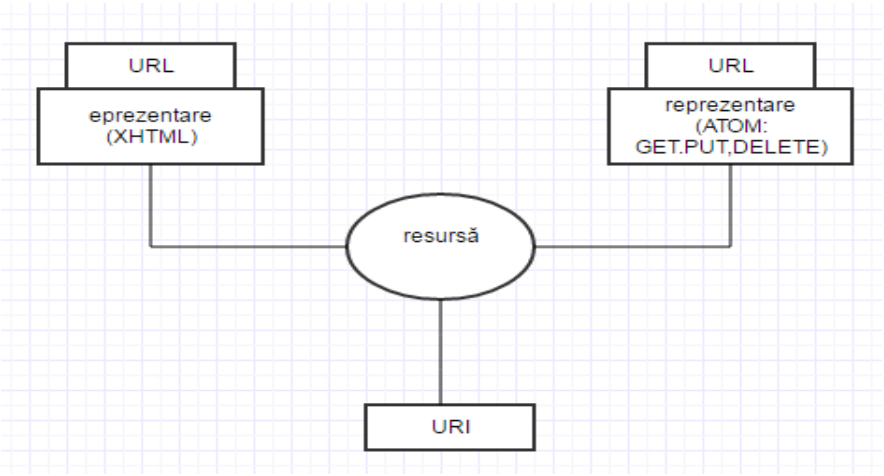
După cum s-a descris, serverul este o componentă distribuită care se apelează remote[35] atunci când este nevoie să se proceseze date pe care acesta le poate administra. Serverul dat este publicat pe platforma de Cloud Heroku unde la fel se află și baza de date cu care acesta este conectat. Serverul este implementat specific arhitecturii REST pentru sistemele distribuite.

[35] Apel la distanță

Conform descrierii prezente în sursa [9] și [19], arhitectura REST este un stil de comunicare între aplicația client și server a unui sistem. Această arhitectură se preferată a fi folosită peste arhitectura SOAP(Simple Object Access Protocol)[36], deoarece arhitectura REST necesită o lățime de bandă mai mare, ceea ce arată ca o potrivire mai bună pentru utilizarea acestei arhitecturi la comunicarea prin Internet. Arhitectura REST, comunică peste protocolul HTTP (Hypertext Transfer Protocol) și are câteva constrângeri arhitecturale conform sursei [7]:

- Decuplarea consumatorului de producător
- Stateless existence – fiecare cerere este considerată independentă
- Capabil să mențină un cache[37]
- Capabil să fie implementat într-un sistem pe mai multe nivele
- Capabil să conțină o interfață uniformă

Cel mai des arhitectura REST este utilizată în aplicații mobile, site-uri web, rețele sociale și procese de afaceri automatizate. Flexibilitatea comunicării este asigurată prin alocarea de resurse unice (URIs[38] – Universal Resources Identifiers). Iar aceasta se datorează metodele HTTP specifice cu semnificații diferite, cum sunt metodele: GET, POST, PUT și DELETE[39]. Rezultatul unei procesări dintre metode, conduce la returnarea reprezentării unei resurse. Fiecare reprezentare a unei resurse are asociat un URL iar reprezentarea datelor este modelată în XML



Figură 5.7 Reprezentarea URI's

[36] Simple Object Access Protocol – protocol de comunicare care permit diferitor sisteme de operare să comunice folosind protocolul HTTP și XML.

[37] Memorie rezervată pentru a o putea accesa la viteză mare

[38] Mecanism folosit pentru accesarea unei resurse

[39] Metode specifice protocolului HTTP care pot fi apelate.

Un exemplu de procesare a resursei este prezentată în codul de mai jos. Acest script, este responsabil de crearea imaginilor cu locația, folosind Google Static Maps și coordonatele geografice primite de la aplicația utilizator. Încărcarea lor se face pe Cloud(Cloudinary), și salvarea referințelor inclusiv a datelor corespunzătoare pentru utilizator în baza de date.

```php
<?php
    require 'Cloudinary.php';
    require 'Uploader.php';
    require 'Api.php';

    $app_data = $_POST['data'];
    $json = json_decode($app_data);
    $positionLat = $json->{"positionLat"};
    $positionLong = $json->{"positionLong"};
    $positionPlace = $json->{"positionPlace"};
    $time = $json->{"time"};
    $user_email = $json->{"user"};

    if(!empty($positionLat) && !empty($positionLong) && !empty($user))
    {
        $image_string_url =
                    $positionLat,$positionLong&sensor=false&key=A5Y";
        $image = file_get_contents($image_string_url);
        $fp = fopen('temp_image.png', 'w+')
        fputs($fp, $image);
        fclose($fp);
        unset($image);
        # Cloudinary credential settings.
       \Cloudinary::config(array(
            "cloud_name" => "ilies-me",
            "api_key" => "171236384335942"
        ));
        # Upload image to the cloud.
        $default_upload_options = array("tags" => "basic_sample");
        $returnedArray = \Cloudinary\Uploader::upload("temp_image.png",
                          $default_upload_option
        # Save data to database
        $conn_string = "dbname=d1s6drrg1op0ji
                        host=ec2-174-129-37-54.compute-1.amazonaws.com
                        port=5432
                        user=lgypruojnuuypc
                        password=NPa4PT-eduh_pserMO_H06ZUCp
                        sslmode=require ";
        $image_url = $returnedArray['url'];
        $conn = pg_connect($conn_string);
        $sql = "INSERT INTO speedings(user_email, user_place,
                                      speeding_time, map_image)
                VALUES ('$user_email','$positionPlace',
                        '$time', '$image_url')";
        $result = pg_query($conn, $sql);
            if ($result)
                print('Succes!');
            else
                print('Insertion Error!');
        pg_close($conn);
    }
    else
        print("Oooopss :)");
?>
```

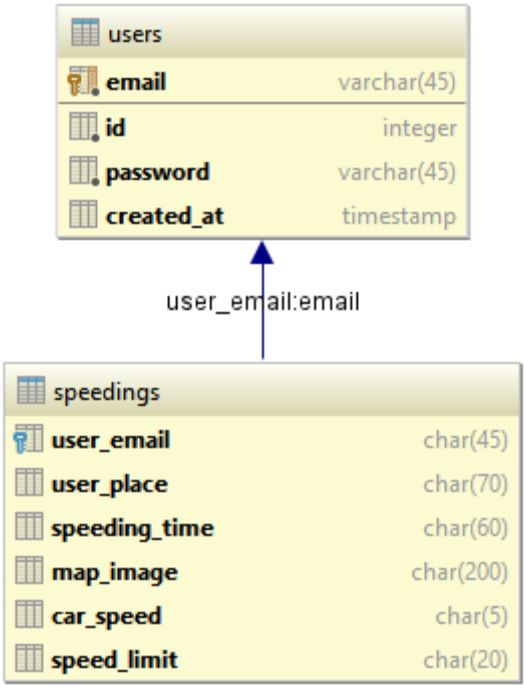
Exemplu de script PHP

5.4. Baza de date PostgreSQL

Sistemul folosește o bază de date pentru a stoca informațiile utilizatorilor și informații despre depășirii limitei de viteză. Chiar dacă baza de date este una care conține doar două tabele, a fost necesară dezvoltarea acestei componente detașat pentru a separa accesul la baza de date de restul aplicației. Ținând cont de constrângerile puse de platforma de Cloud pe care baza de date este publicată(Heroku). Heroku ne obligă să folosim PostgreSQL. PostgreSQL este un sistem de management al bazelor de date relaţionale open-source, care are ca scop principal să devină un standard, care încearcă să adopte standardele ANSI/ISO [40]SQL împreună cu alte extensii.

Comparată cu alte baze de date relaţionale, PostgreSQL se diferenţiază de ele printr-un suport avansat în ceea ce priveşte orientarea pe obiecte, suport pentru tranzacţii sigure, atomicitate, consistenţă, izolare, durabilitate(ACID). Din cauza tehnologiei cu care a fost dezvoltată această bază de date, PostgreSQL este foarte capabilă la manipularea eficientă a mai multor sarcini în acelaşi timp. Rezolvarea concurenţei se realizează fără blocări ale tabelelor(locks), datorită implementării mecanismului de Multiversion Concurrncy Control(MVCC).

Digrama bazei de date.



Figură 5.8 . Structura bazei de date

Pentru această structură avem o relație de tip 1-N(one-to-many)[41]. În tabela „users” avem detaliile care identifică un utilizator în sistem. În cea de-a doua tabelă, tabela cu

[40] American National Standards Institute și International Organization for Standardization
[41] Relație specifică bazelor de date relaționale. Legătură Unu-la- mai Mult

numele “speedings” avem detaliile despre încălcările făcute de utilizator. Un utilizator poate avea mai multe încălcări, de-aici apare și relația “unul-la-mai mulți”. În cea de-a doua tabelă se găsesc informații precum: locația utilizatorului, timpul detectării la care s-a detectat depășirea de viteză, viteza de deplasare a autovehiculului, viteza maximă permisă de deplasare și adresa URL a imaginii în care se specifică locația care a fost salvată în Cloud în acel moment.

Capitolul 6. Testare şi Validare

Testarea aplicaţiei este o măsură a asigurării calităţii(Quality Assurance). Testarea presupune evaluarea calităţii productului realizat pentru a-l îmbunătăţi în caz că se găsesc potenţialele probleme sau erori. Se identifică mai multe tipuri de testări, atunci când vorbim despre o aplicație mobilă, dintre care: testare Front-End[42], testare Back-End,[43] testare resurse(cum ar fi rutele), testarea la stres, testarea compatibilității, testarea securităţii, testarea performanței, testarea automată.
Majoritatea testelor trebuie să asigure că funcţionalitățile cerinţelor aplicaţiei sunt realizate şi funcționează corect.

Front-End Testing
Un prim pas al verificării UI-ului a fost verificarea corectitudinii scripturilor HTML împreună CSS-ul. S-a verificat și corectitudine formelor HTML existente prin validări ale valorilor introduse împreună cu detectarea acelor valori care au fost introduse incorect.
Pentru testarea codului JavaScript s-a folosit Qunit[44].

Back-End Testing
Pentru testarea funcţionalității serverului, s-au realizat Unit-teste folosind frameworkul PHPUnit[45], pentru a garanta eficiența scriptorilor elaborate în limbajul PHP. Tot aici s-a făcut şi teste ce priveşte suprasolicitarea serverului, pentru a se vedea cum se comportă serverul atunci când mai mulţi clienţi apelează serviciile din acesta. Fiind aleasă o soluţie Cloud, problema de suprasolicitare a serverului s-a dovedit a fi una rezolvată.

Testarea la condiții de stres
Acest tip de testare testează cum se comportă aplicația în momentul unor condiții anormale. De exemplu, specific pentru sistemul dat este intervalul de timp cu care resursele serverului sunt apelate de către aplicația client. Din cauza a unor factori precum este acoperirea limitată a rețelei de telefonie, primirea unui rezultat la apelarea unor resurse poate fi întârziată. A fost nevoie de testat acest aspect, și de realizat o soluție optimă în așa fel încât sistemul să revină cât mai repede la funcționarea normală.

Testarea compatibilității
Fiind realizată o aplicație hibridă, s-a mers din start pe ideea că aplicația trebuie să fie compatibilă cu toate platformele existente. Însă, se poate întâmpla ca o anumită versiune de plugin care se găsește în Cordova să aibă suport pentru iOS şi să nu aibă suport pentru Android. În acest caz este nevoie să se investigheze acest caz și să se implementeze undele condiții în fișierul de pre-configurare Cordova. Un alt aspect este legat de browser-ele dispozitivelor. Unele navigatoare web suportă o anumită abordare a implementării, altele nu.

[42] Aplicație utilizator
[43] Aplicație care rulează pe partea de server
[44] Framework destinat pentru testarea interfeței utilizator
[45] Framework destinat pentru testarea aplicațiilor PHP

Testare performanței

Testarea performanței se rezumă la cât de repede aplicația prelucrează o cerere atunci când aceasta se solicită. În aplicațiile mobile, timpul de așteptare după momentul când s-a tastat un buton este aproximativ 300ms. După testarea și cercetarea unor posibilități de optimizare a performanței în cadrul aplicațiilor hibride, s-a găsit o librărie AngularJS, care se numește ngTouch, și face ca timpul de așteptare în momentul în care se face procesează un eveniment să fie redus.

Pe lângă testele de mai sus, cel mai des s-a utilizat testarea manuală a sistemului, specific mai ales pentru aplicația client. Pentru aplicațiile mobile, cea mai relevantă metodă de testare este testare pas cu pas a cerințelor funcționale ale aplicației. Cel mai bun feedback[46] oferit în cadrul testării vine de la utilizatori, întrucât ei sunt cei ce folosesc cel mai des aplicația. Mai jos sunt detaliate unele scenarii de test care s-au folosit pentru detectarea posibilelor erori și operații nefuncționale.

Caz de testare 1: Înregistrare unui cont nou în sistem
Descriere: Utilizatorul trebuie să poată să se înregistreze în sistem cu ajutorul unei adrese de email sau a unui număr de telefon împreună cu o parolă.
Scenariul 1:
Pasul 1 : Se oprește conexiunea la Internet.
Pasul 2 : Se pornește aplicația
Pasul 3 : Se apasă butonul *"Register"*
Rezultat așteptat: Un mesaj care sugerează o eroare din cauza lipsei de conexiuni la Internet este afișat.

Scenariul 2:
Pasul 1 : Se acordă conexiunea la Interent dispozitivului.
Pasul 2 : Se pornește aplicația
Pasul 3 : Se apasă butonul *"Register"*
Pasul 4 : Se alege *„Register with mail"* și se apasă *"Create Account"*
Rezultat așteptat: Butonul *„Create Account"* este inactiv, întrucât câmpurile obligatorii nu au fost completate

Scenariu 3:
Pasul 1: Se pornește aplicația
Pasul 2: Se apasă butonul *"Register with mail""*
Pasul 3: Se introduce o adresă de email invalidă
Pasul 4: Se apasă *"Create Account"*
Rezultat așteptat: Butonul *„Create Account"* este inactiv, deoarece câmpul de Email nu a fost validat.

Scenariu 4:
Pasul 1: Se pornește aplicația
Pasul 2: Se apasă butonul *"Register"*
Pasul 3: Se introduce o adresă de email validă iar restul câmpurilor goale
Pasul 4: Apasă „*Creează cont*"

[46] Reacție cu scopul de menținere a echilibrului în sistemele informatice

Rezultat așteptat: Butonul *„Create Account”* este inactiv, deoarece câmpul de Email nu a fost validat.

Caz de testare 2: Detectare vitezei de deplasare și a limitei de viteze
Descriere: Viteza de deplasare și viteza maximă admisă trebuie să fie afișată
Scenariul 1:

Pasul 1 : Se oprește serviciul “Locație dispozitiv”
Pasul 2 : Se pornește aplicația(Accesul la Internet trebuie să fie permis)
Pasul 3 : Se introduc credențialele și se apasă butonul *“Login”*
Pasul 3 : Pagina de unde trebuie să se afișeze viteza de deplasare și limita de viteză apare.
Rezultat așteptat: Un mesaj care sugerează o eroare din cauza că serviciu de locație a dispozitivului este deconectat apare.

Scenariul 2:

Pasul 1 : Se pornește serviciul “Locație dispozitiv”
Pasul 2 : Se pornește aplicația(Accesul la Internet trebuie să fie permis)
Pasul 3 : Se introduc credențialele și se apasă butonul *“Login”*
Pasul 4 : După ce s-a autentificat în sistem cu succes, se oprește conexiunea la Internet
Rezultat așteptat: Un mesaj care sugerează o eroare din cauza că conexiunea la Internet a dispozitivului este deconectată apare.

Caz de testare 3: Vizualizarea locațiilor unde s-a încălcat limita de viteză.
Descriere: Utilizatorul trebuie să poată vede locațiile unde limita de viteză a fost depășită.
Scenariu:

Pasul 1 : Se oprește conexiunea la Internet.
Pasul 2 : Se pornește aplicația
Pasul 3 : Se autentifică în aplicație
Pasul 4 : Se navighează spre pagina cu istoric, apăsând butonul *“Show History”*
Rezultat așteptat: Un mesaj care sugerează o eroare din cauza lipsei de conexiuni la Internet este afișat.

Capitolul 7. Manual de Instalare si Utilizare

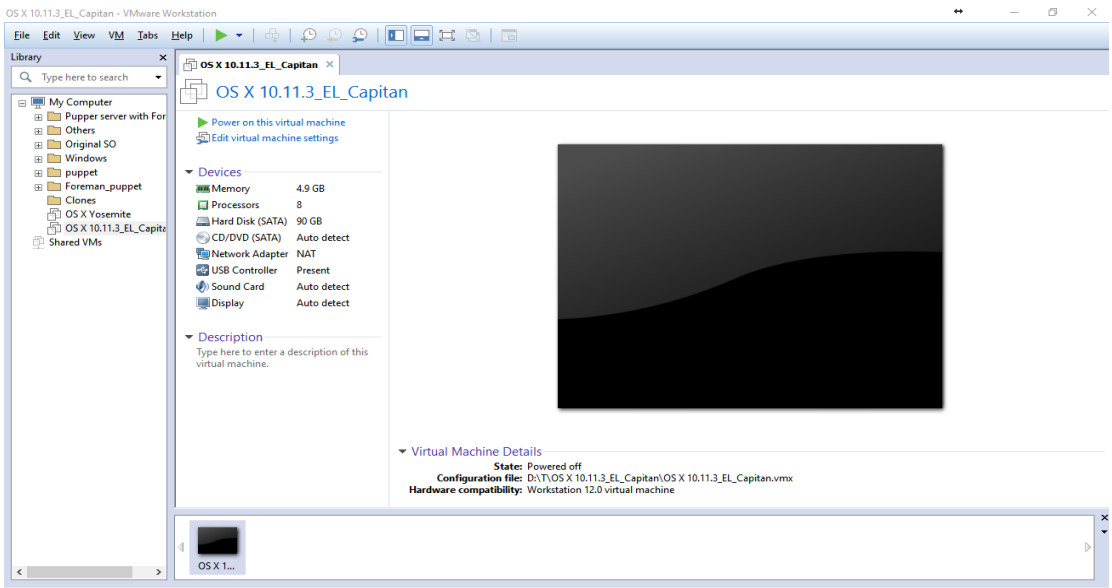
În această secțiune sunt detaliate resursele software și hardware necesare pentru instalarea și rularea aplicației, precum și o descriere pas cu pas a procesului de instalare. Tot aici, pentru cei care doresc să modifice acest sistem, sau să continue această implementare, sunt descriși pașii de instalare și configurare a mediului de dezvoltare, împreună cu tehnologiile folosite.

Aplicație client, cu denumirea "Speed Limit Assistant" va putea fi găsită în toate magazinele de aplicații existente pentru fiecare platformă în parte. Va fi îndeajuns să se navigheze în acel magazin de aplicații, precum [AppStore, WindowsStore, GooglePlay][47], să se caute aplicația cu denumirea dată și să se instaleze.
Odată ce aplicația a fost instalată, aceasta va apărea pe ecranul dispozitivului de unde va putea fi accesată și folosită. În imaginea de mai jos se identifică aplicația instalată pe un dispozitiv care rulează iOS.

Pentru instalarea aplicației offline. Tutorial de instalare pentru dispozitivele iOS.

Pentru instalarea offline a aplicației pentru acest tip de dispozitive, este nevoie să se folosească mediul de programare oferit de către compania Apple – xcode[48]. Acest tool poate fi folosit doar pe sistemele de operare OSx[49], (noua denumire Mac OS).

Dacă se folosește un dispozitiv MacBook, e simplu, însă în cazul în care nu avem un astfel de dispozitiv, e nevoie să se folosească o mașină virtuală cu un astfel de sistem de operare. Pentru dezvolarea sistemului s-a folosit o mașină virtuașă cu sistemul de operare El Capitan, versiunea OS X 10.11.3 rulat cu ajutorul hipervizorului[50] VMware Workstation.



Figură 7.1 Configurațiile mașinii virtuale

[47] Magazine de aplicații specifică unei anumite platforme
[48] Mediu de programare(IDE) Apple
[49] Sistem de operare pentru calculatoarele personale Apple
[50] Software de virtualizare care prezintă un sistem hardware emulat.

După configurarea și rularea cu succes a mașinii virtuale, este nevoie să se configureze și mediul de programare împreună că tool-urile necesare pentru dezvoltare. Un prim pas este descărcare și instalarea xcode IDE din magazinul de aplicații AppleStore. După ce acest pas a fost îndeplinit, este necesar să se instaleze Apache Cordova împreună cu Ionic. Mai jos sunt pașii de instalare.

În terminalul sistemului de operare se itroduc pe rând următoarele comenzi.

```
$ sudo brew install node
```

```
$ sudo npm install -g cordova
```

```
$ sudo npm install -g ionic
```

După rularea comenzilor de mai sus, tot ce a mai rămas să se facă este să se pregătească aplicația pentru a putea fie instalată pe telefon. Pentru asta este nevoie în primul rând de codul sursă a acesteia care se află pe CD, în arhiva Speed-Limit-Assistant.zip. Se dezarhivează arhiva, și se merge cu linia de comandă a terminalului la locația codului sursă.

```
$ cd Speed-Limit-Assistant
```

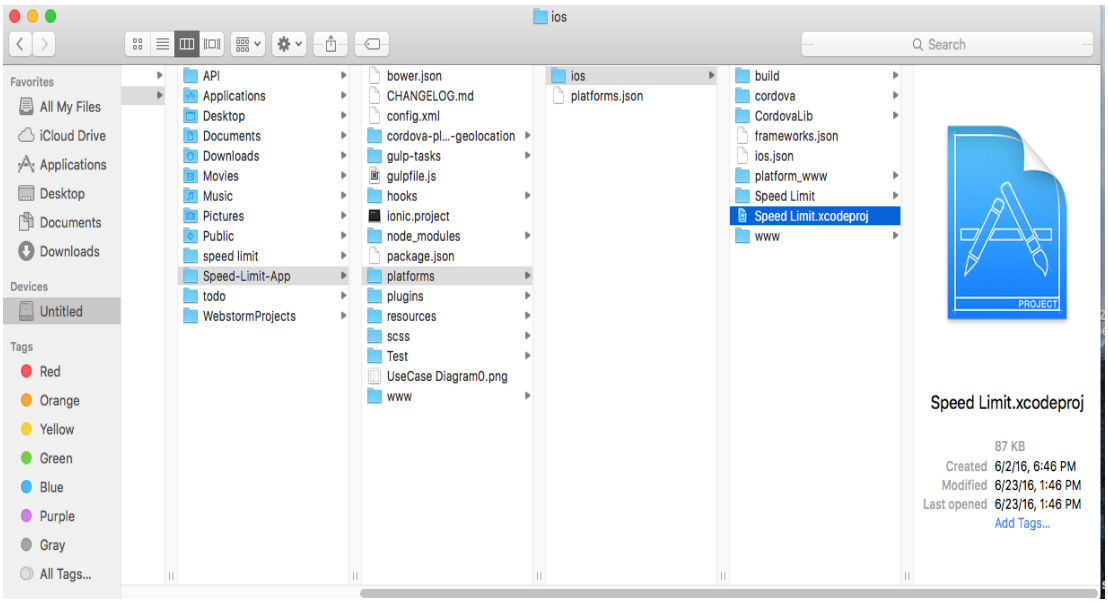
Pasul următor este să se adauge platforma necesară pe care această aplicație trebuie să ruleze. În cazul dat este necesar să se adauge platforma iOS însă dacă se dorește instalarea aplicației pe o altă platformă, trebuie specificată platforma aleasă, de exemplu Android, WindowsPhone etc.

```
$ sudo ionic platform add ios
```

Odată ce platforma dorită a fost adaugată, aplicația trebuie compilată pentru platforma specifică în modul următor,

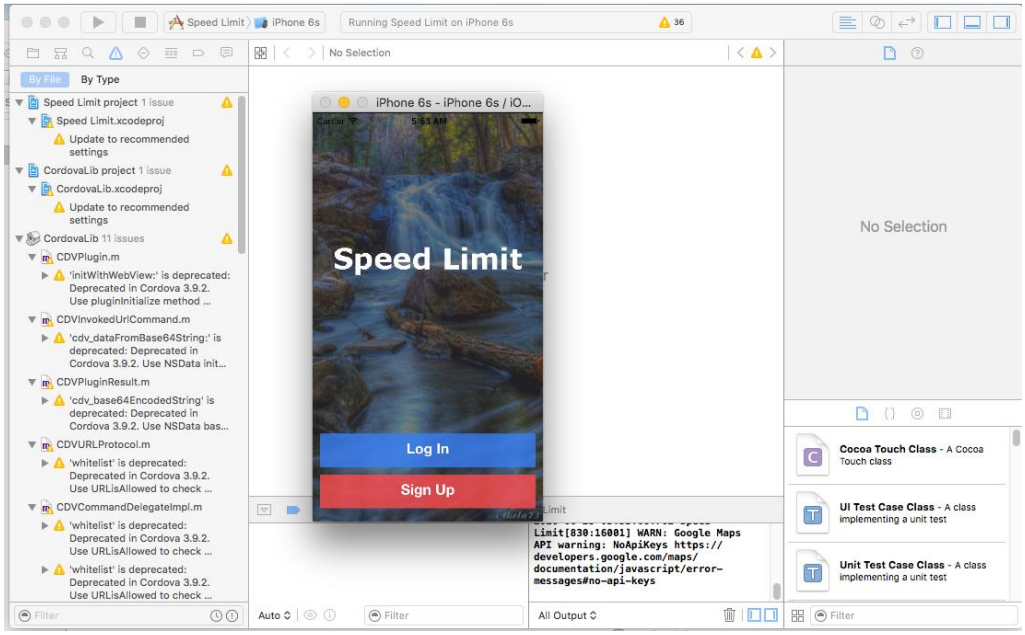
```
$ sudo ionic build ios
```

După executare pașilor de mai sus, aplicația va fi configurată cu toate setările necesare, iar în arborele de fișiere vor fi adăugate noi directoare și fișiere specifice pentru platforma aleasă, după cum se pot vedea în figura de mai jos.



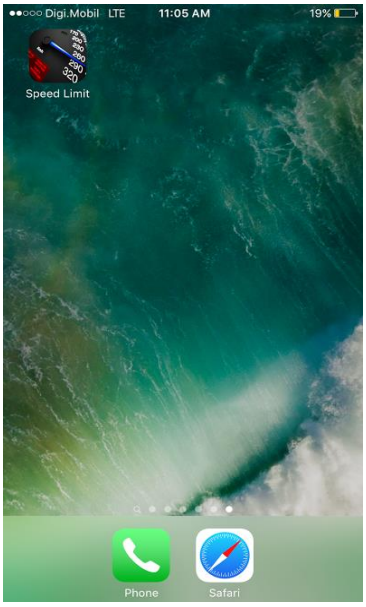
Figură 7.2 Locație fișier cu extensia .xcodeproj

Executând fișierul Speed-Limit.xcodeproj, se va deschide mediul de programare xcode, care ne permite instalarea aplicației pe dispozivit sau într-un simulator[51]. Pentru instalarea aplicației pe dispozitivul fizic, este necesar doar de conectarea prin cablul USB al acestuia la portul calculatorului și să se aleagă din lista disponibilă de dispozitive din xcode acel dispozitiv, apoi să se tasteze butonul Run.



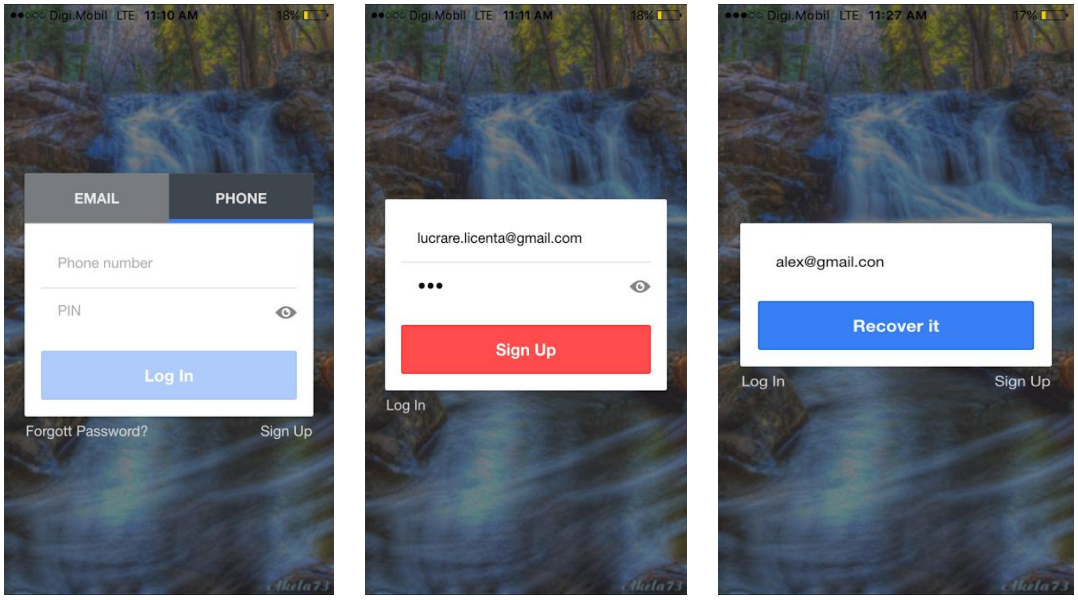
Figură 7.3 Aplicație rulată în simulator

[51] Dispozitiv virtual pe care se poate rula aplicații



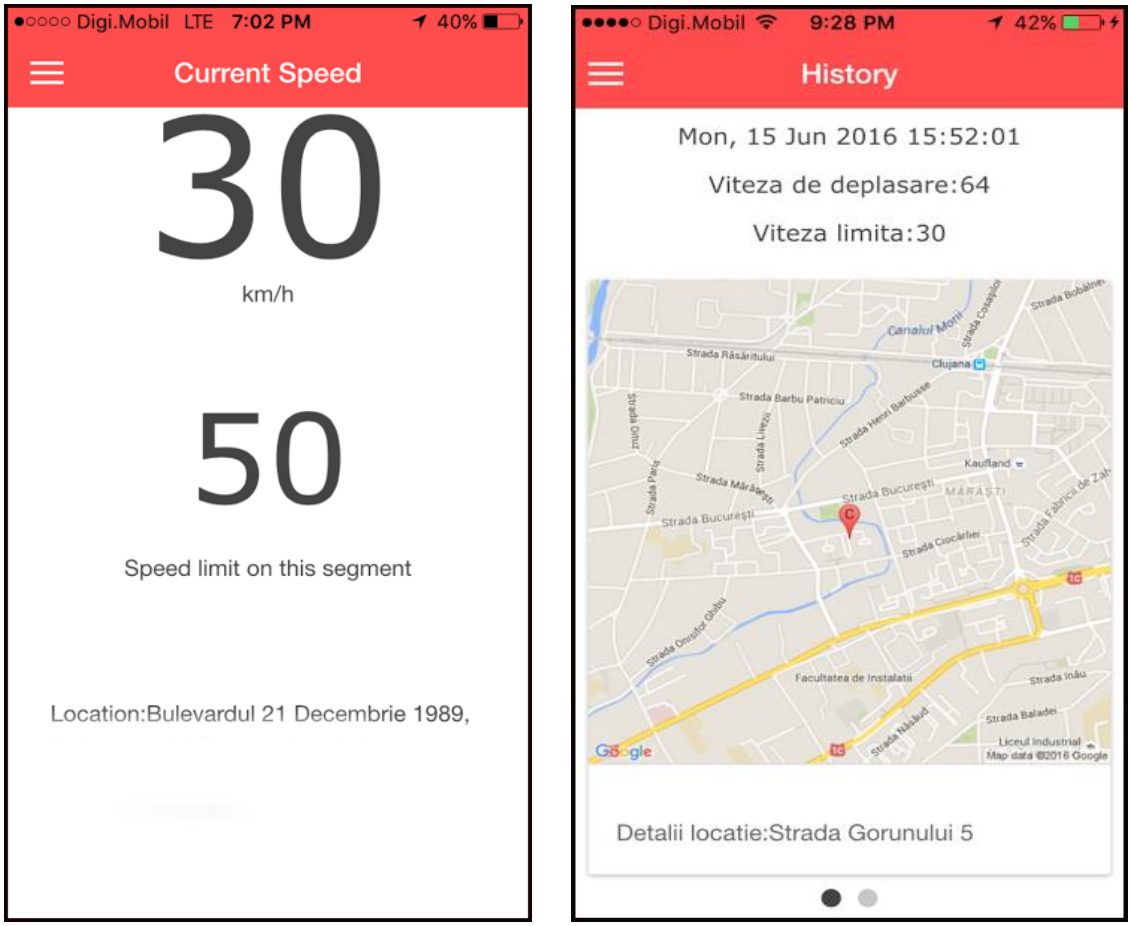
Figură 7.4 Imagine aplicație instalată pe dispozitiv fizic real

La pornirea aplicației, utilizatorul poate alege crearea unui cont sau autentificarea sa în cazul în care este deja înregistrat. În figura de mai jos se evidențiază pagina de înregistrare și pagina de autentificare în aplicație. În caz că utilizatorul a fost înregistrat deja și solicită recuperarea credențialelor în caz că au fost uitate.



Figură 7.5 Pagina de înregistrare și autentificare și recuperare credențiale.

La autentificare cu succes în aplicație, va apărea pagina în care se va indica viteaza de deplasare și viteza maximă admisă pentru locația unde se află la momentul dat autoturismul, cu se poate vede și in figura următoare



(A) (B)

Figură 7.6 Imaginea A - pagina de afișaj a vitezei de deplasare, locația și viteza limităde deplasare.
Imaginea B - istoric al încălcărilor. Cuprinde data, ora, viteza de deplasare, viteza limită și locația.

AICI MAI AM DE AGĂUGAT INFORMAȚIE

Capitolul 8. Concluzii

În acest capitol sunt prezentate realizările, obiectivele care s-au atins prin acest proiect, dar și descrierea dezvoltărilor și îmbunătățirilor viitoare.

8.1. Realizarea obiectivelor propuse

Sistemul care s-a realizat reușește să își atingă scopul, de a fi un sistem cât mai practic, ușor de folosit și cel mai important utilă pentru utilizatorii care-l folosesc.

Prin intermediul acestui proiect, s-au dobândit cunoștințe noi și s-au pus în practică informații studiate la diferite materii pe parcursul anilor universitari.

Pentru dezvoltarea sistemului s-a parcurs următoarele etape:

- S-au studiat tehnologiile care pot fi folosite pentru implementare.
- S-au dobândit cunoștințe solide despre tehnologiile hibride și conceptele pe care acestea funcționează.
- S-a studiat piața, și anume serviciile de Cloud care pot fi utilizate și serviciile web oferite de unele companii cum sunt Google și HERE.
- S-au analizat sistemele similare.
- S-a dezvoltat un plan de dezvoltare a aplicației. Detalii cu cerințele și obiectivele planului se pot găsi la adresa de mai jos. https://trello.com/b/MuYANFG5/driver-speed-limit-dsl
- S-a reușit să se pună în aplicare toate obiectivele principale și secundare ale sistemului, cum sunt:
  - Înregistrare utilizator
  - Autentificare utilizator
  - Detectarea vitezei de deplasare
  - Detectarea vitezei maxime permise
  - Atenționarea utilizatorului când acesta depășește limita de viteză
  - Salvarea locației, data, ora când s-a detectat o depășire a vitezei limite
  - Posibilitatea de a vedea un istoric al depășirilor limitei de viteză
  - Trimiterea statisticilor pe mail la sfârșit de fiecare săptămână

8.2. Dezvoltări ulterioare

După lansarea aplicației și atingerea unui succes de 1000 utilizatori înregistrați, se va dori o creștere. Acest lucru fiind motivat de ideea de a fi cât mai aproape de utilizator și de a putea satisface cât mai multe necesități. La fel se va urmări și extinderea capacităților aplicației, pentru a deveni cât mai complexă, sigur din punct de vedere funcțional cât și non-funcțional. Acest lucru se va putea realiza numai și numai prin interacțiunea cu utilizatorul și punere accent pe doleanțele acestuia. Iar o cale eficientă și care nu necesită scheme elaborate – chestionarul. Realizat conform informațiilor pe care se va dori cunoscute – chestionarul va fi cel mai bun element de cercetare a pieții. Cu toate acestea, deja există câteva modificări sau chiar funcționalități noi. Un comportament aparte ar fi - adăugarea expresiei de muzică. Prin implementarea acestei

funcționalități, se dorește să se obțină existența unui playlist de muzică, specializat pe individual de către utilizator. Pe lângă acest playlist, care poate fi interpretat ca și muzică statică (având în vedere că muzica se va încărca pe server de către utilizator și nu va putea fi eliminată, până în momentul în care utilizatorul va crede de cuviință), se va opta și pentru funcționalitatea de a adăuga o listă formată din 2-3 posturi de radio cel mai des ascultate de șofer. Prin această realizare se dorește să se două avantaje majore, care îl va face pe utilizator să se simtă cât mai confortabil posibil. Prima ar fi – selectarea unor cântece, în funcție de parte a zilei în care șoferul va conduce. Altfel spus, se ține cont de siguranța șoferului și se cunoaște că noaptea – este o perioadă când accidentele ating o cotă mai mare, iar motivul principal este admorirea la volan. Pentru că acest lucru nu se dorește a întâmpla, muzica standard, pusă de către dezvoltator și celelalte adăugate de utilizator, va avea un efect pozitiv – îl va ține pe șofer treaz. A doua calitate – radio, se urmărește ca totul să fie la îndemână șoferului. El să nu mai fie nevoit, când urcă la volan, să își deschidă și aplicația (pentru securitate) și să își caute postul de radio favorit (pentru divertisment). Când se va implementa această funcționalitate, utilizatorul prin deschiderea aplicației va face defapt două acțiuni care îi vor satisface nevoia de securitate cât și cea de divertisment. În așa mod, șoferul va câștiga un mare beneficiu – timp, iar aplicația – mai multă vizibilitate și utilitate în viața utilizatorului.

O a doua funcționalitate care se va dori implementată este o platformă, în cadrul aplicației care va conține știrile care țin de domeniul traseele naționale cât și de cele județene, municipale. Se consideră utilă, pentru că îl pune pe șofer mereu cu un pas înaintea schimbărilor și din timp își va putea construi traseul pe drumul cel bun. Tot aici va primi și știri ce țin de modificarea legislației șoferului sau care sunt cerințele pentru acesta. Având în vedere că aceste știri vor fi selectate și vor fi puse la îndemâna utilizatorului, putem vorbi despre un grad mai înalt de utilizare a aplicației, ceva inovativ, eficient și scutire de timp.

A treia caracteristică și nu cea din urmă – o platformă în care instituțiile publice autorizate care se ocupă de drumuri, legislația ce privește șoferul, vor putea interacționa cu șoferul. Lunar, instituția va putea pune în platformă, un chestionar din care va vrea să afle anumite aspecte ce îi vor ajuta să implementeze legi, să realizeze activități cât mai eficiente. Respectiv, prin această funcționalitate, se va ajuta interacțiunea amiabilă dintre cetățean – șofer și instituția publică pentru realizarea unor lucruri frumoase și în primul rând lucruri care se vor întâmpla pentru și în favoarea șoferului.

Cele trei proiecte menționate mai sus, vor putea fi modificate pe parcurs pentru îmbunătățire. Dar, odată cu ele se mai doresc următoarele funcționalități care se vor adăuga cu succes lângă cele prezente. Acestea ar fi:

- Adăugarea unui navigator real-time
- Salvarea unui istoric legat de drumul parcurs împreună cu , începând cu locația de start și locația finală.
- Posibilitatea de a vedea locațiile sau porțiunile de drum în care s-au depistat cele mai multe încălcări de viteză
- Consumarea unor servicii ce oferă statistici privind segmentele de drum cu cele mai multe accidente.
- Implementarea unui mecanism de autoînvățare a sistemului. Aceasta prevede posibilitatea utilizatorului de a alege un drum (de exemplu drumul care acesta îl face de la locul de muncă spre casă cu mașina) și de a stoca

local date despre limita de viteză. Asta ar reduce consumul traficului de date pe Internet.

Bibliografie

[1] **A. Froehlich**, Smartphone OS: A 22-Year History, 17 Martie 2015, http://www.informationweek.com/mobile/mobile-applications/smartphone-os-a-22-year-history/d/d-id/1319495

[2] **Omaima Bamasak, Ghadah Al-Mehmadi, Amal Al-Ghami** Monitored Intelligent Car Speed Adaptation (MISA) System http://www.ijens.org/vol%2011%20i%2003/113203-7575%20ijecs-ijens.pdf

[3] **R. Nixon**, Learning PHP, MySQL, Javascript, and CSS, O'Reilly Media, Inc., 27 August 2012

[4] **M. Sarwar and T. R. Soomoro**, Impact of Smartphone's on Society, European Journal of Scientific Research, ISSN 1450-216X / 1450-202X Vol.98 No 2 Martie 2013, pp. 216-226,

[5] **Rachel Appe**l, Modern Apps: Mobile Web Sites vs. Native Apps vs. Hybrid Apps. https://msdn.microsoft.com/en-us/magazine/dn818502.aspx

[6] **Sofia Sundstrom**, Responsice design in mobile app, http://www.sthlmconnection.se/en/blog/why- responsive-design, [Accesat 05, Mai 2015]

[7] **R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach și T. Berners-Lee** „Hypertext Transfer Protocol -- HTTP/1.1," [Interactiv]. Available: http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf.

[8] **Kennet Vuong,** Player monitoring system, University of Oslo, Master's Thesis.

[9] „Web Services Architecture," 11 Februarie 2004. [Interactiv]. Available: http://www.w3.org/TR/ws-arch/wsa.pdf. [Accesat 11 Octombrie 2013].

[10] **Sunil Arora**, Best hybrid API frameworks. http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5-js-css

[11] **Kristopher Sandoval** Best Hybrid Mobile App UI Frameworks http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5-js-css

[12] **Andy Gup** How accurate is GPS http://www.andygup.net/html5-geolocation-api-%E2%80%93-how-accurate-is-it-really/

[13] Documentația oficială Ionic http://ionicframework.com/docs/ http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5-js-css

[14] Documentație oficială Apache Cordova https://cordova.apache.org/docs/en/latest/guide/overview/

[15] **Irene Ros** REST API Documentation Best Practices https://bocoup.com/weblog/documenting-your-api

[16] **M. Mulikita**, Mobile Application Testing, Master Thesis of Science at University of Applied Sciences Cologne

[17] **Ionuț Anghel** ,Curs PHP
http://users.utcluj.ro/~ianghel/DAW/curs/2016_DAW_Curs5_PHP.rar
[18] **Ionuț Anghel** ,Curs AJAX
http://users.utcluj.ro/~ianghel/DAW/curs/2016_DAW_Curs4_AJAX_JQUERY.rar
[19] **Ionuț Anghel** ,Curs Servici Web
http://users.utcluj.ro/~ianghel/DAW/curs/2016_DAW_Curs9_WS.rar
[20] Documentația oficială **Cloudinary**. http://cloudinary.com/documentation
[21] Documentația oficială **Heroku** https://devcenter.heroku.com/
[22] Documentația oficială **Cloud9** https://docs.c9.io/docs
[23] Documentația oficială AngularJS https://docs.angularjs.org/api
[24] Detalii de funcționare Wase https://www.waze.com/ro/

Referințe figuri

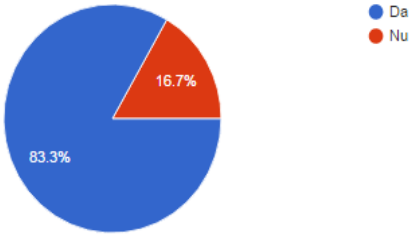
Referință tabele

Întrebările și rezultatul studiului realizat.

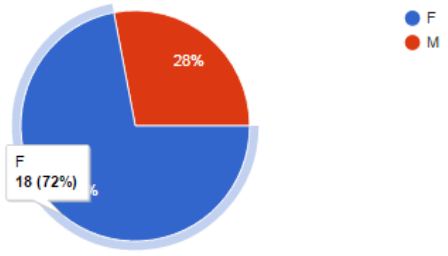
Număr de participanţi:31

Detalii oferite fiecărui individ care a răspuns la chestionar. Sistemul dat este o aplicație care este implementată pentru SmartPhone fără/cu un SmartWatch. Detaliile esențiale sunt: la un interval de timp predefinit(3, 5 secunde) aplicația primește informații precum: numele/denumirea străzii/segmentului de drum pe care automobilul se află și viteza maximă permisă pe acel segment de drum. Având aceste date, în funcție de viteza de deplasare a automobilului, aplicația comunică cu un server unde transmite date precum: locația, viteza de deplasare, viteza maximă permisă pe acel segment de drum. În caz că viteza a fost depășită sunteți atenționați cu o notificare pe telefon sau SmartWatch. Mai mult ca atât aveți și un raport care vă va salva toate încălcările pe care le-ați avut, de când s-a instalat aplicația.

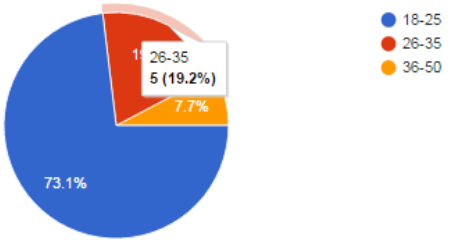
Sunteți șofer? (dacă răspunsul este nu, da-ți submit la chestionar și vă mulțumim)



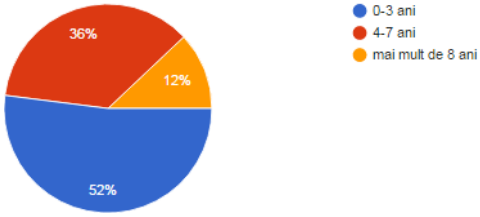
Gen



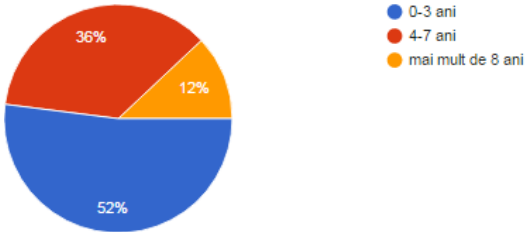
Vârsta



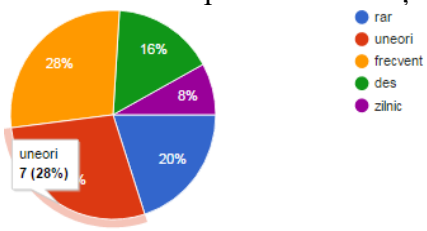
De cât timp aveți permis de conducere?



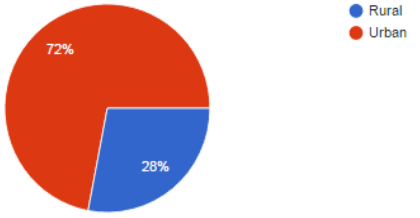
De cât timp șofați?



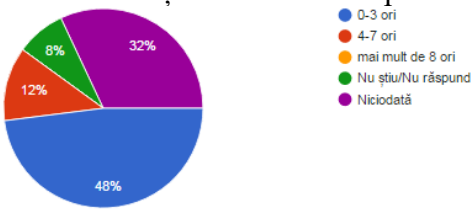
Cât de des în săptămână utilizați mașina?



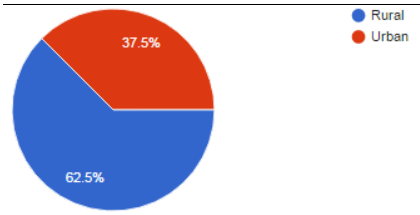
Mediul preponderent unde conduceți mașina?



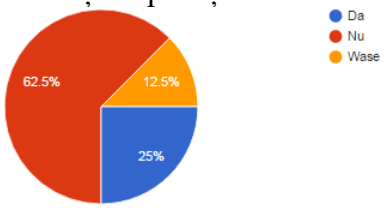
De câte ori ați fost amendat/ă pentru încălcarea limitei de viteză, în ultimele 12 luni?



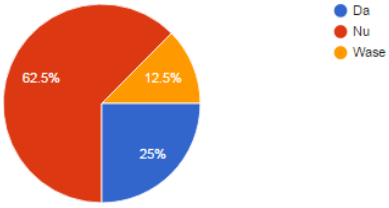
Cel mai des ați încălcat viteză în mediul:



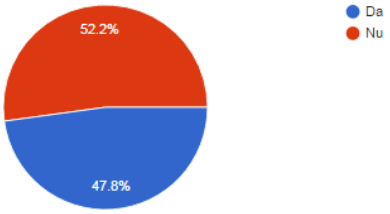
Utilizați o aplicație asemănătoare cu Speed Assistant? (dacă nu, săriți peste întrebare)



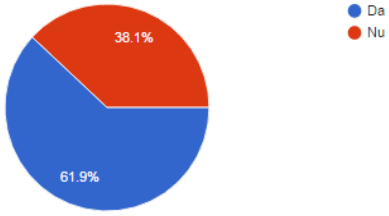
Dacă da, acea aplicație oferă date precum limita de viteză și raport cu viteza încălcată?



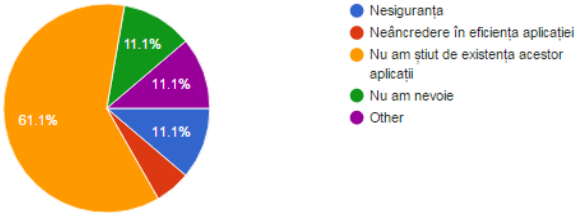
Ați folosi sistemul „Speed assistant"?



Credeți că este util acest sistem pentru dvs?



Dacă nu ați utiliza aplicația noastră, care este motivul?



Ce așteptări aveți de la această aplicație?

- Foarte bune

- Sa îmi dea senzația de siguranță
- Sa fie eficientă
- Ar fi interesant să vezi limitele de viteză
- Acuratețe, feedback rapid, să dea alertă când încalc viteza.
- Sa îmi fie util
- Să mă ajute sa fiu atentă la limita de viteza
- În timp scurt să știu unde si când am încălcat
- Să se ridice la așteptările mele, să aibă și o interfață accesibil de utilizat
- O să utilizez această aplicație pentru a vedea care sunt avantajele acesteia față de ceea ce am

Acest chestionar a fost realizat cu scopul de a cerceta un pic piața și de a se vedea dacă șoferii sunt predispuși să se încreadă în aplicații de genul Speed Assistant. Grupul țintă a fost format din persoane care conduc o mașină, respectiv au permis de conducere .

S-a utilizat 15 întrebări de diferite tipuri – atât deschise cât și închise, pentru a oferi persoanei care completează chestionarul, capacitatea de a se expune cât mai natural și de a sustrage cât mai multe informații relevante pentru o aplicație.

Prima categorie de întrebări a ținut cont de identitatea persoanei (cu mențiunea că datele vor fi confidențiale). A doua categorie de întrebări a ținut de setarea locului unde conduce mașina, numărul de sancțiuni pe care le-a primit și mediul în care cel mai des se află. A treia categorie de întrebări combină informații legate de utilizarea unor astfel de aplicații, și care sunt așteptările exacte pe care le are față de acest sistem. Cum se observă, cuprinde o arie complexă care deține informații importante legată de piață.

Din 31 de răspunsuri, au răspuns 83,3 % care s-au declarat că sunt șoferi și respectiv 16.7% că nu sunt. Au răspuns la acest chestionar bărbați în proporție de 28% și femei – 72%. Mai mult decât atât s-a reușit să se interogheze toate grupele de vârstă setate inițial – 73.1 % persoane cu vârstă între 18 și 25 ani; 19.2% persoane cu vârstă între 26 și 35 ani și în proporție de 7.7% persoane cuprinse între 36 și 50 de ani.

Totodată, s-a aflat ceva poate mai puțin obișnuit în legătură cu perioada de când persoana intervievată are permis de conducere și de când timp șofează. Pentru a vedea diferența, cel mai bine e ca rezultatele să fie afișate în oglindă.

Așadar, persoanele care au permis de conducere în intervalul 0-3 ani au fost 52% și tot în același grup de ani, se încadrează și 64% care șofează. A doua categorie de ani 4-7, înregistrează un procentaj de 36% - deținător de permis de conducere și că tot din acei 83.3% care s-au declarat că sunt șoferi, conduc în proporție de 32%. Ultima categorie, adică cei care au permis de conducere mai mult de 8 ani și respectiv conduc de mai mult de opt ani, este în proporție de 12%. S-a observat că cei care au fost eligibili pentru acest chestionar, conduc zilnic în proporție de 8%, rar – 20% iar celălalt procentaj este repartizat pentru utilizarea mașinii de la uneori, frecvent până la des. Mediul în care este utilizată mașină, în proporție de 72% este urban iar 28 la sută rural. Tot aici se poate face legătura cu întrebarea legată de unde cel mai des s-a încălcat viteza – mediul rural – 62.5% și 37.5 procente în mediul urban. S-a observat că în mediul rural (mediul rural însemnând drumurile naționale, din afara orașului) este mai periculos din motiv că viteza e la ordinea zilei și respectiv încălcările sunt mai des întâlnite. Și pe de altă parte, în oraș șoferii dau dovadă de o atenție sporită – fie din cauza prezenței autorității care menține ordinea sau că traficul este la o cotă ridicată – la fel aceste teorii merită de analizat și

interpretat. Revenind la scopul acestui chestionar – dacă șoferii sunt reticenți sau ba în utilizarea unei aplicații care i-ar notifica când se încalcă limita de viteză. Și aici este adevărul – 82.6 la sută utilizează o aplicație asemănătoare cu cea pe care s-a prezentat dar 62.5 la sută din ei spun că acea aplicație totuși nu le prezintă date precum limita de viteză și raport cu viteza încălcată. Ultima relație ține de reticența sau deschiderea clientului de a utiliza acest sistem și motivul pentru care nu ar utiliza aplicația și respectiv care sunt așteptările pe care le au de la aplicație (sigur, pentru cei care au răspuns că vor utiliza sistemul). În proporție de 47.8% ar folosi acest sistem, având și așteptări destul de logice. Răspunsurile au fost grupate în funcție de categoriile bine delimitate – majoritatea dintre ei își doresc ca sistemul să fie eficient, util și să cunoască detalii în momentul întâmplării acțiunii, iar o altă categorie își doresc acuratețe în prezentarea datelor, sunt interesați de raportul despre care s-a vorbit la începutul prezentării și 2 dintre ei au argumentat alegerea prin prisma interesului trezit. Procentajul care urmează a fi prezentat nu este atât de dezamăgitor ci mai mult provocător, în ideea de a se putea convinge și cealaltă parte – 52.2% să utilizeze un asemenea sistem. Pentru a veni în apărarea lor, motivele invocate ar fi: nesiguranța, neîncrederea în eficiența aplicației și că nu au nevoie.

În concluzie: piața care vinde astfel de aplicații e în dezvoltare, necesitatea există și oamenii sunt conștienți de acest fapt. Ce ar trebui să facă un producător de acest tip de produs, e să analizeze cât mai bine cerințele și așteptările potențialului client și să țină în pas cu noile tehnologii și nu în ultimul rând să pună accent pe informarea clientului cu orice noutate legată de aplicație.