Mendelova univerzita v Brně
Provozně ekonomická fakulta

---

# Carfriend – webová aplikácia pre hlásenie nežiadúcich situácií vzťahujúcich sa k vozidlám

**Bakalárska práca**

Vedúci práce:
Ing. Pavel Turčínek, Ph.D.

Tomáš Jakúbek

Brno 2016

Ďakujem vedúcemu bakalárskej práce, Ing. Pavlovi Turčínkovi Ph.D., za metodické vedenie, ochotu, čas a cenné rady, vďaka ktorým som mohol dokončiť tento text. Poďakovanie patrí taktiež priateľom, ktorí asistovali pri testovaní aplikácie.

**Čestné prehlásenie**

Prehlasujem, že som prácu: **Carfriend – webová aplikácia pre hlásenie nežiadúcich situácií vzťahujúcich sa k vozidlám**
vypracoval samostatne a všetky použité zdroje a informácie uvádzam v zozname použitej literatúry. Súhlasím, aby moja práca bola zverejnená v súlade s § 47b zákona č. 111/1998 Sb., o vysokých školách v znení neskorších predpisov a v súlade s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Som si vedomý, že sa na moju prácu vzťahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzatvorenie licenčnej zmluvy a použitie tejto práce ako školského diela podľa § 60 odst. 1 Autorského zákona.
Ďalej sa zaväzujem, že pred spísaním licenčnej zmluvy o použití diela inou osobou (subjektom) si vyžiadam písomné stanovisko univerzity, že predmetná licenčná zmluva nie je v rozpore s oprávnenými záujmami univerzity a zaväzujem sa uhradiť prípadný príspevok na úhradu nákladov spojených so vznikom diela, a to až do ich skutočnej výšky.

V Brne dňa 18. mája 2016 .....

## Abstract

Jakúbek, T. Carfriend – web application for reporting unpleasant situations related to vehicles. Bachelor thesis. Brno, 2016.

This bachelor thesis deals with the development of mobile application with web technologies and simultaneously describes tools and technique for hybrid application development. Carfriend application is designed for road users. They create content by sending mutual notifications and messages using vehicle's unique evidence plate numbers. Notifications alert, report and predict unpleasant situations, that vehicle may face. This paper describes connection and communication with server service Backand, which uses NoSQL interface to save data.



## Keywords

Web application, HTML, CSS, AngularJS, Apache Cordova, BaaS, hybrid development, traffic app



## Abstrakt

Jakúbek, T. Carfriend – webová aplikácia pre hlásenie nežiadúcich situácií vzťahujúcich sa k vozidlám. Bakalárska práca. Brno, 2016.

Táto bakalárska práca sa zaoberá tvorbou mobilnej aplikácie využitím webových technológií a súbežne popisuje nástroje a postup tzv. hybridného vývoja. Aplikácia Carfriend je svojou povahou určená pre účastníkov cestnej premávky. Tí tvoria obsah posielaním vzájomných upozornení a správ prostredníctvom jedinečných evidenčných čísiel vozidiel. Upozornenia varujú, oznamujú a predpovedajú vodičom nepríjemné situácie, ktorým vozidlo môže čeliť. Práca popisuje prepojenie a komunikáciu so serverovou službou Backand, ktorá ukladá dáta pomocou NoSQL rozhrania.



## Kľúčové slová

Webová aplikácia, HTML, CSS, AngularJS, Apache Cordova, BaaS, hybridný vývoj, dopravná aplikácia

# Obsah

# 1 Úvod a cieľ práce

## 1.1 Úvod do problematiky

Oblasť dopravy je stále obľúbeným námetom pre mobilné a webové aplikácie. Môže za to hlavne jej veľký rozsah a priamy vplyv na množstvo ľudí. Účastníkom cestnej premávky sa stáva vedome každý, kto vstúpi už len na chodník. Z toho dôvodu táto oblasť skrýva bohaté množstvo potenciálnych užívateľov aplikácií. Vznikajúce aplikácie sa snažia priniesť úžitok hlavne vodičom. Či už tým, že im hľadajú parkovanie, navigujú ich alebo informujú o dopravnej situácii. Často sa ale zabúda na spolucestujúcich, chodcov a ďalších účastníkov. Tí neraz disponujú informáciou určenou pre vodičov ale nevedia ako im ju jednoducho a rýchlo predať.

Len samotné zaparkovanie vozidla na ulici môže majiteľovi spôsobiť niekoľko nečakaných nepríjemností. Od odtiahnutia vozidla, poškodenia až po jeho krádež. Žiadneho majiteľa vozidla tieto situácie netešia a určite by sa im najradšej vyhol. Tu vzniká nápad vytvoriť nástroj, ktorý by zabezpečil prenos informácií medzi jednotlivými účastníkmi cestnej premávky. Jednoduché riešenie, ktoré by prepojilo aspoň časť účastníkov spočíva v používaní evidenčného čísla vozidla ako identifikátor konkrétnej osoby. Prepojiť ľudí pomocou evidenčných čísel je nový a výnimočný spôsob komunikácie, ktorý, ako bude uvedené v tejto práci, otvára veľa možností.

Keby existoval všadeprítomný nástroj, ktorým môže očitý svedok alebo poškodená osoba majiteľa okamžite upozorniť alebo ešte lepšie varovať, nemuseli by napríklad odťahovacie služby v Brne v roku 2014 odviezť až 15 tisíc vozidiel, a tým spôsobiť vodičom až 30 miliónovú škodu (Boček, Šulek a Cibulka, 2014). Takýmto nástrojom je jednoznačne internet. Internet poskytuje prostredie na komunikáciu, na vykonanie akcií sú však potrebné aplikácie. Je zrejmé, že také množstvo účastníkov cestnej premávky nie je možné pokryť hocijakým typom aplikácie.

Aplikácia, ktorá nie je na ničom závislá a je dostupná všade tam, kde je internet sa nazýva webová. Uvedené nahráva webu obrovskú výhodu a potenciál, a preto patrí webový prehliadač medzi najpoužívanejšie počítačové programy. Pre mobilných užívateľov však nie je prehliadač až takým šťastným riešením. Dnes už ale ani na nich nie sú webové technológie prikrátke a v tejto práci bude predvedené, že dokážu toho omnoho viac.

Webové technológie vedia byť tak prefíkané, že ich mobilný užívateľ používa a pritom o tom ani sám nevie. Dokážu splynúť s natívnym mobilným prostredím a zároveň sa distribuovať medzi všetkými mobilnými platformami. Preto je ich možné použiť na vývoj falošných mobilných aplikácií. Prívlastok falošný sa v literatúre nahrádza pojmom hybridný, ktorý zhŕňa v sebe to hlavné z mobilného a webového sveta. Hybridné webové aplikácie alebo skrátene hybridy, v súčasnosti vo veľkom obsadzujú obchody s mobilnými aplikáciami, a to z dôvodu jednoznačne ľahšieho a lacnejšieho vývoja.

## 1.2 Cieľ práce

Táto práca si dáva za cieľ navrhnúť a implementovať dopravnú aplikáciu spájajúcu všetkých účastníkov cestnej premávky prostredníctvom rýchlych upozornení. Práca sa zaoberá tvorbou hybridnej mobilnej aplikácie použitím webových technológií, ktoré komunikujú so službami cloudového servera. Práca je teda zameraná na využitie moderných spôsobov a trendov ako aj v oblasti vývoja aplikácií, tak aj v otázke serverových možností.

Na aplikáciu sú kladené požiadavky jednoduchosti, moderného vzhľadu, ktoré sú ďalej doplnené funkcionalitou zistenou dotazníkovým šetrením. Po preštudovaní kľúčových technológií práca prinesie hotové návrhy aplikácie, zvolí vhodné implementačné nástroje a v závere zhodnotí výsledok, vytkne nedostatky a navrhne možné rozšírenia.

# 2 Prehľad literatúry

## 2.1 Webová aplikácia

Webová aplikácia je program, ktorý je uložený na vzdialenom serveri a cez internet je prenášaný klientovi do prehliadača (Wikipedia, 2015). Remick (2011) definuje webovú aplikáciu ako aplikáciu, ktorá využíva web a webové technológie na vykonanie jednej alebo viacerých úloh cez sieť. Medzi webové technológie zaraďuje JavaScript, HTML[1], CSS[2], Javu a ďalšie programovacie jazyky určené na použitie v sieti. Sieťou Remick (2011) myslí internet alebo intranet.

Webové aplikácie sa neinštalujú na klientskych počítačoch, a tým nezapĺňajú lokálne úložiská a aj preto je ich obľúbenosť neustále na vzostupe. Nakoľko na spustenie takejto aplikácie je potrebný len prehliadač, tak webové aplikácie je bezproblémové používať na rôznych operačných systémoch. Ich ďalšou výhodou je vždy aktualizovaný stav.

### Tradičná webová aplikácia

V tradičných (serverovo-orientovaných) webových aplikáciách je HTML kód generovaný na strane servera a posielaný do prehliadača. Pre každú žiadanú stránku je vždy vytvorená nová HTTP[3] požiadavka na server, ktorý ju vykoná a naspäť odošle klientovi HTTP odpoveď (Mitchel, 2013). To následne spôsobí, že celá stránka je nahradená novou a okno prehliadača viditeľne preblyskne. V tomto prístupe je zaťaženie na klientskej strane minimálne a JavaScript je prevažne používaný na tvorbu jednoduchých zmien v užívateľskom prostredí alebo na tvorbu animácií (Fink a Flatow, 2014).

Fink a Flatow (2014) považujú za častý problém v tomto type webových aplikácií situáciu, kedy serveru trvá spracovanie požiadavky príliš dlho, aplikácia nereaguje a užívateľ musí čakať. Táto pomalá skúsenosť sa stáva neprijateľnou pre čím ďalej vyše sofistikovaného webového užívateľa (Mikowski a Powell, 2014). Navyše, bez internetového pripojenia nie je možné serverovo-orientovanú aplikáciu vôbec používať, pretože práve na server smerujú všetky dátové a stavové dotazy a ten nie je bez internetu dostupný (Fink a Flatow, 2014).

### Jednostránková aplikácia

Jednostránková aplikácia z angl. Single Page Application (niekedy tiež nazývaná One Page Application) je webová aplikácia s jednou HTML stránkou, ktorá sa raz inicializuje a pri ďalších interakciách sa mení len potrebný obsah (Monteiro, 2014). Takmer celá takáto aplikácia prebieha priamo v klientskom prehliadači, zatiaľ čo server slúži väčšinou len na autentifikáciu a ako zdroj, či úložisko dát (Fink a Flatow, 2014).

[1]HTML - Hyper Text Markup Language
[2]CSS - Cascading Style Sheets
[3]HTTP - Hyper Text Transfer Protocol

V porovnaní so serverovo-orientovanými aplikáciami je hlavný rozdiel spôsob prijímania a odosielania dát po prvotnej HTTP požiadavke. Jednostránkové aplikácie využívajú AJAX[4] na prenos dát medzi serverom a klientom, ktoré sú vo formáte JSON[5] (Fink a Flatow, 2014). To znamená, že v momente, keď dáta dorazia ku klientovi, klient čiastočne prerenderuje HTML stránku bez nutnosti opakovaného obnovenia celej stránky. Práve preto nedochádza k sťahovaniu celého HTML kódu, čo sa odzrkadlí v rýchlosti SPA aplikácií. V prípadoch, kedy aplikácia *musí* čakať na odozvu servera, tak upozorní užívateľa animovanými indikátormi priebehu (Mikowski a Powell, 2014).

Nakoľko je prevažná časť funkcionality a logiky aplikácie na klientskej strane, tak prevládajúcim programovacím jazykom v tomto type aplikácií je JavaScript. Práve preto Monteiro (2014) upozorňuje, že nie je možné spustiť SPA aplikáciu v prehliadačoch s chýbajúcim alebo vypnutým JavaScriptom. V niektorých jednostránkových aplikáciách môže byť problematický aj dlhší čas prvotného načítania z dôvodu sťahovania všetkých potrebných skriptov.

AJAX a dynamická tvorba webu na strane klienta je veľkým problémom pre vyhľadávacích robotov (Mrozek, 2013). Tí, na rozdiel od prehliadačov, nedokážu spustiť a vykonať JavaScript, a preto nemajú prístup k celému obsahu na stránke. To má za následok, že hlavný obsah, ktorý je skrytý v JavaScripte, robot nebude indexovať, a teda ho nebude možné ani vyhľadať (Fink a Flatow, 2014). Z tohto dôvodu je tento prístup vhodný skôr pre aplikácie ako pre textové firemné stránky (Mrozek, 2013).

## 2.2 Natívna aplikácia

Natívna aplikácia je spustiteľná aplikácia, ktorá musí byť na danom zariadení najskôr nainštalovaná. Môže bežať len na jednej platforme–platforme, pre ktorú bola vytvorená–a to je jej najväčšou silou, ale zároveň aj slabinou. (Fink a Flatow, 2014)

Natívne aplikácie sú navrhnuté tak, aby využívali výhody lokálneho hardwaru a operačného systému. Práve táto závislosť zvyšuje náročnosť vývoja týchto aplikácií, pretože aplikácia závisí nielen od architektúry zariadenia, ale aj od typu, verzie operačného systému a nainštalovaných ovládačov (Fink a Flatow, 2014). Existencia veľkého množstva rôznych operačných systémov zapríčiňuje nedostupnosť natívnej aplikácie pre všetkých potenciálnych užívateľov.

Natívne aplikácie môžu využívať aj lokálne databázy a interakcia s nimi je o mnoho rýchlejšia ako používanie vzdialených serverových zdrojov. Ďalšou ich výhodou je dostupnosť a funkčnosť aj bez internetového pripojenia. (Fink a Flatow, 2014)

Natívna mobilná aplikácia je naprogramovaná v jazyku, ktorý je priamo podporovaný operačným systémom zariadenia. Natívnym jazykom je najčastejšie Java pre Android zariadenia, Objective-C pre operačný systém iOS a C# pre Windows

[4] AJAX - Asynchronous JavaScript and XML
[5] JavaScript Object Notation

Phone. Natívny vývoj je špecifický pre každú mobilnú platformu a vyžaduje použitie vývojárskych nástrojov, ktoré nemusia byť podporované na každom operačnom systéme. Nasledujúca tabuľka sumarizuje hlavné závislostí vývoja natívnych aplikácií.

Tabuľka 1: Závislosti natívneho vývoja

| **Platforma** | **Jazyk** | **Vývojové prostredie** | **Požadovaný OS** | **Obchod** |
|---|---|---|---|---|
| iOS | Objective-C | Xcode | Mac OS | App Store |
| Android | Java | Eclipse | Mac OS, Windows | Google Play |
| Windows Phone | C#, C++ | Visual Studio | Windows | Windows Store |

Natívne aplikácie majú plný a priamy prístup k natívnym funkciám a rozhraniam z čoho pramení množstvo výhod. Medzi tie patrí dosiahnutie najlepšej úrovne prevedenia a výkonu aplikácie, možnosť špecializácie vývojárov na jednu platformu, jednoduchšia a hlbšia integrácia s hardwarom zariadenia, najvyššia zabezpečenosť aplikácií, najlepší užívateľský zážitok, viacdotykové gestá (dvojklik, priblíženie), natívne prvky užívateľského prostredia s plynulými animáciami a peňažný profit umiestnením koncovej aplikácie do natívnych obchodov (Wilken a Bradley, 2015).

Natívny vývoj je náročnejší a jeho uplatnenie vyžaduje zvyčajne viac času a skúseností, čo sa následne odzrkadľuje vo vyšších nákladoch.

## 2.3 Hybridná aplikácia

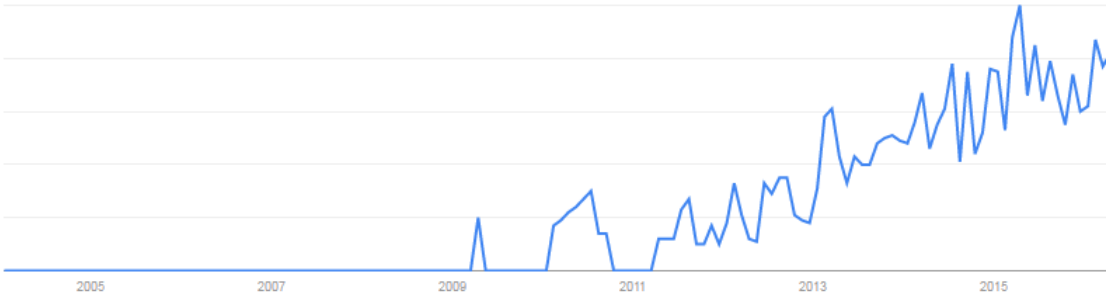
Hybridná (multiplatformová) aplikácia je najčastejšie (existujú aj iné spôsoby, ale tento je najpopulárnejší) postavená na webových technológiách, ale na rozdiel od webovej stránky má prístup k natívnemu API[6] daného zariadenia. Je teda akýmsi hybridom medzi webom a natívnou aplikáciou, ktorý sa snaží zmixovať to najlepšie z obidvoch prístupov. (Panhale, 2016)

Vývojárka Appel (2014) vníma hybridnú aplikáciu ako skvelý spôsob premiestnenia obsahu z webových stránok do obchodov s aplikáciami, a teda aj do samotných mobilných zariadení. Podľa spomenutej vývojárky môže publikácia hybridnej aplikácie slúžiť ako spúšťač marketingového procesu, zatiaľ čo sa skutočná natívna aplikácia len vyvíja.

Hybridný vývoj je založený na koncepte *„build once, run everywhere“* a oproti natívnemu vývoju umožňuje prenositeľnosť rovnakého kódu medzi takmer všetky platformy. Táto výhoda podstatne zľahčuje vývoj a znižuje jeho náklady, a to je hlavný dôvod, prečo sa hybridný vývoj stáva čoraz populárnejším (Panhale, 2016).

Graf na obrázku 1 znázorňuje záujem o pojem *hybrid mobile application* od roku 2005 až po súčasnosť. Hodnoty na grafe vyjadrujú počet vyhľadávaní daného výrazu vzhľadom na celkový počet vyhľadávaní v službe Google za celý čas, a teda nepredstavujú absolútne množstvo vyhľadávania (Google Trends, 2016). Z grafu je možné jednoznačne interpretovať rastúci záujem o tieto aplikácie.

[6] Aplikačno-programové rozhranie




Obrázok 1: Záujem o pojem *hybrid mobile application* (Google Trends, 2016)

Hybridné aplikácie sa inštalujú rovnako ako natívne a užívateľ ich taktiež spúšťa z domovskej obrazovky. Internetové pripojenie nie je nutnosťou na spustenie a používanie hybridných aplikácií.

Najväčšou slabinou hybridov je výkon, ktorý v porovnaní s natívnymi aplikáciami býva zvyčajne nižší a limitovaný kvalitou WebView (Panhale, 2016). Hybridná aplikácia je taká výkonná ako aj inštancia WebView (Wilken a Bradley, 2015) a z toho dôvodu je možné pocítiť rozdiel v rýchlosti najmä na starších zariadeniach (Kovács, Sivák 2013). Druhou slabou stránkou je náročnosť napodobnenia natívneho vzhľadu a užívateľského rozhrania. Pri hybridnom vývoji je potrebné sa grafikou približovať k natívnej grafike, pretože napríklad Apple zamieta aplikácie vo svojom obchode, ktoré nespĺňajú normy týkajúce sa užívateľského rozhrania.

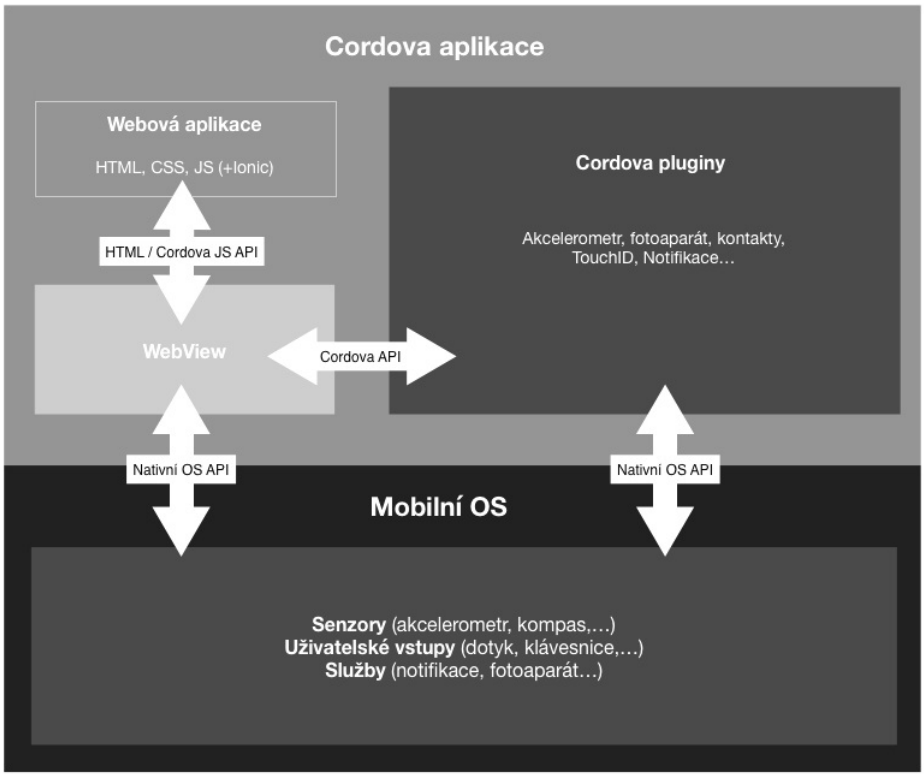
Ako ukážkový príklad hybridnej aplikácie sa uvádza aplikácia Netflix. Aj Facebook aplikácia na platformu iOS bola v roku 2011 vyvinutá najskôr ako hybridná aplikácia, avšak neskôr ju Facebook nahradil natívnou. Dôvodom bolo pomalé renderovanie užívateľského rozhrania. (Panhale, 2016)

### WebView

Appel (2014) vo svojom článku uvádza, že hybridná aplikácia je *zapuzdrovač* webovej stránky. V skutočnosti je naozaj takáto aplikácia doslova zabalená do natívneho kontajnera (tzv. natívny wrapper). Práve natívny wrapper poskytuje premostenie k natívnym funkciám ako je napríklad fotoaparát alebo telefónny zoznam. Tento fakt je možné sledovať na obrázku 2, ktorý okrem iného popisuje architektúru hybridov.

Keď je takáto aplikácia spustená, wrapper najskôr vytvorí inštanciu prehliadača, ktorá sa zvyčajne nazýva WebView, a tá následne zobrazí webovú aplikáciu (Wilken a Bradley, 2015). Vďaka tomu, že WebView nemá typické ovládacie prvky webových prehliadačov, hybridná aplikácia je pre užívateľa na prvý pohľad nerozlíšiteľná od natívnej.

WebView je systémový prvok, ktorý je vopred nainštalovaný v mobilných zariadeniach a umožňuje aplikáciám zobrazovať webový obsah (Google Inc., 2016). Komponent WebView pracuje na technológii predvoleného prehliadača a je špeci-




Obrázok 2: Architektúra hybridnej aplikácie (Václavík, 2015)

fický pre každú platformu. Panhale (2016) uvádza delenie jednotlivých WebView v závislosti na operačnom systéme:

- Android -> WebView -> technológia Chrome -> WebKit engine,
- iPhone -> UIWebView -> technológia Safari -> Webkit engine,
- Windows phone -> WebView ->technológia IE -> Trident engine.

Komunikáciu medzi WebView a natívnym prostredím zabezpečujú nástroje ako napríklad Adobe PhoneGap (Apache Cordova) alebo Trigger.io. Tieto nástroje nie sú súčasťou zariadení, ale jedná sa o nástroje tretích strán. Hybridné frameworky[7], ako bývajú spomínané nástroje nazývané, sú tvorené množstvom malých JavaScriptových knižníc a funkcií, prostredníctvom ktorých dokáže vývojár komunikovať s väčšinou základných natívnych funkcií zariadenia.

### Runtime

Hybridné aplikácie typu runtime majú vlastné exekučné prostredie, ktoré umožňuje vyvíjať natívnu aplikáciu napríklad v čistom programovacom jazyku JavaScript. Najväčším predstaviteľom tohto spôsobu tvorby multiplatformovej aplikácie

[7]Aplikačný rámec

je framework Appcelerator Titanium. Titanium poskytuje kód, ktorý je napísaný v natívnom jazyku cieľovej platformy. Zdrojový kód aplikácie sa kombinuje práve s týmto natívnym kódom. Po spustení aplikácie nedochádza k obaleniu aplikácie do natívneho wrapperu, ale k interpretácii kódu JavaScriptovým enginom. (Powell, 2015)

#### Kompilátory

Používanie kompilátorov vyžaduje skúsenosti najpokročilejších vývojárov. Zdrojový kód aplikácie je kompilovaný buď do tzv. medzikódu, natívneho jazyka cieľovej platformy alebo do strojového kódu. Tento prístup sa často kombinuje s predchádzajúcim a výsledkom je komplexná natívna aplikácia s vysokým výkonom pre každý operačný systém. Nástroj presadzujúci tento prístup je napríklad Xamarin. (Xamarin, 2015)

## 2.4 Porovnanie typov aplikácií

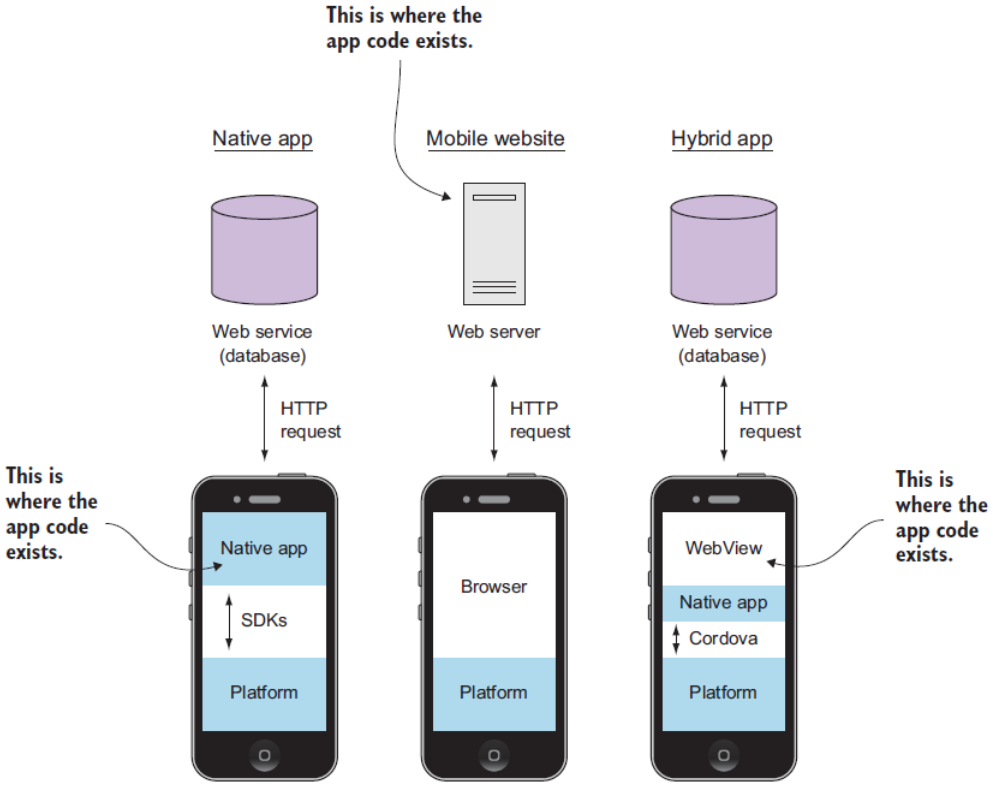
Každá architektúra aplikácie prináša vývojárom a používateľom určité výhody a je vhodná pre špecifické prípady. Táto podkapitola sumarizuje rozdiely natívnej, webovej a hybridnej aplikácie v grafickom a slovnom vyjadrení.

Obrázok 3 porovnáva spomenuté tri typy aplikácií z hľadiska dizajnu a architektúry. Na tomto obrázku je takisto znázornený prístup k databáze alebo webovým službám. Šípka ukazuje na miesto, kde je uložený kód aplikácie. Jediné, čo majú v tomto porovnaní všetky tri typy spoločné, je komunikácia so serverom pomocou protokolu HTTP.

Tabuľka 2 porovnáva typy aplikácií z hľadiska zvolených charakteristík. V tomto zrovnaní je možné nájsť vzájomné prieniky, ktoré potvrdzujú postavenie hybridov medzi natívnym a webovým vývojom.

Tabuľka 2: Porovnanie vlastností natívneho, webového a hybridného vývoja

| | **Typ aplikácie** | | |
|---|---|---|---|
| **Charakteristika** | **Natívna** | **Webová** | **Hybridná** |
| Prístup k natívnemu API | áno, plný | nie | áno, neúplný |
| Distribúcia | natívne trhoviská | web | natívne trhoviská |
| Prenositeľnosť medzi platformami | nie | áno | áno |
| Výkon | veľmi rýchla | pomalá | rýchla |
| Grafika | natívna | CSS, SVG | CSS, SVG |
| Vývojárske schopnosti | Objective-C, Java, C# | HTML, CSS, JS | HTML, CSS, JS |
| Pripojenie | online a offline | zväčša online | online a offline |




Obrázok 3: Porovnanie architektúr natívnej, webovej a hybridnej aplikácie (Wilken a Bradley, 2015)

## 2.5 JavaScript

JavaScript je v dnešnej dobe základným kameňom pri tvorbe moderných webových aplikácií (Suehring, 2015). Podľa Pehlivaniana a Nguyena (2012) je JavaScript mocný, všestranný a všadeprítomný programovací jazyk, keďže ako jediný sa nachádza na takmer všetkých osobných počítačoch po celom svete. Pehlivanian a Nguyen (2012) ďalej definujú JavaScript ako skriptovací jazyk pre web. Tri hlavné vlastnosti JavaScriptu sú zhrnuté v nasledujúcich bodoch (Zakas (2012); Kovalyov (2015); Pehlivanian a Nguyen (2012)):

- JavaScript je interpretovaný jazyk, ktorý musí byť pri každom spustení prekonvertovaný do strojového kódu.
- JavaScript je dynamicky typovaný (alebo slabo typovaný) jazyk, ktorý nevyžaduje explicitne deklarovať dátový typ u premenných a tie teda môžu niesť akékoľvek dáta.
- Syntax JavaScriptu je založená na jazyku C a ďalších podobných jazykoch ako Java a Perl. V názve a v syntaxi podobnosť JavaScriptu a Javy končí.
- Ostatné vlastnosti je možné vyhľadať v publikáciách zmienených autorov.

Hlavnou úlohou JavaScriptu bolo poskytovať interakciu užívateľom. Tá sa využije napríklad pri vyplňovaní formulárov a následnej validácie zadaných vstupov. Dnes je JavaScript používaný pre viac než len vstupné operácie. Umožňuje vývojárom vytvárať celé aplikácie (SPA aplikácie), ktoré konkurujú v rýchlosti a funkcionalite desktopovým aplikáciám. (McPeak, 2015)

JavaScript, ako aj ďalšie skriptovacie jazyky, je implementáciou štandardu ECMAScript. Ten je v dobe písania tejto bakalárskej práce v novej verzii 6 (označovaná ako ES6 alebo Harmony) a jeho podpora v moderných prehliadačoch neustále rastie (Mozilla Developer Network, 2015). Zatiaľ čo ECMAScript je len špecifikácia jazyka, JavaScript je jeho konkrétna implementácia a pre poskytnutie ďalších funkcií a možností webovým vývojárom je nutné mu doplniť ešte dva ďalšie diely – objektový model dokumentu a objektový model prehliadača (Zakas, 2012).

#### Objektový model dokumentu (DOM)

Podľa McPeaka (2015) je DOM jedným z najviac nepochopených štandardov, ktoré ustanovil World Wide Web Consortium. Zakas (2012) definuje model DOM ako aplikačno-programové rozhranie pre XML a HTML dokumenty. Tie sú tvorené hierarchiou uzlov, ktoré budujú stromovú štruktúru a jednotlivé uzly je možné mazať, pridávať, nahradzovať a meniť. Snaha o prístup do určitej časti modelu DOM skôr, ako ju webový prehliadač stihne zostaviť, vedie k chybám (Pehlivanian a Nguyen, 2012).

#### Objektový model prehliadača (BOM)

BOM model poskytuje metódy zaisťujúce komunikáciu s prehliadačom. Nie len JavaScript je objektovo-orientovaný, ale aj prehliadač je tvorený z množstva objektov. Napríklad, v prehliadači existuje objekt `window`, ktorý predstavuje okno prehliadača a viditeľnú oblasť stránky. Známymi metódami tohto objektu sú metódy `alert()` a `prompt()`.

## 2.6 Hybridné nástroje

Ako bolo spomenuté v prechádzajúcej kapitole, hybridné nástroje sú nástroje tretích strán, ktoré zabezpečujú komunikáciu medzi aplikáciou a zariadením. Vývojár, vďaka ním, nemusí ovládať programovacie jazyky jednotlivých operačných systémov a aj napriek tomu je schopný vytvoriť aplikácie, ktoré je možné umiestniť do natívnych obchodov. (Panhale, 2016)

Na dnešnom trhu je možné nájsť množstvo rôznych takýchto nástrojov, ktoré sa odlišujú hlavne podporou platforiem, systémom pluginov[8], spôsobom zapuzdrovania, ladiacimi nástrojmi a emulátormi. V rámci tejto kapitole budú bližšie charakterizované dva hybridné frameworky využívajúce WebView na prístup k natívnemu API, a to Adobe PhoneGap a Trigger.io.

[8]V tomto kontexte je plugin modulom, ktorý poskytuje JavaScriptové rozhranie na prístup k natívnym komponentom.

### 2.6.1 Adobe PhoneGap

Projekt PhoneGap pochádza z firmy Nitobi, ktorá ho predstavila v roku 2008 a išlo o prvý multiplatformový nástroj vôbec. O tri roky neskôr bola táto firma odkúpená spoločnosťou Adobe Systems a spolu s ňou aj projekt PhoneGap. Celý projekt bol následne darovaný firme Apache Software Foundation, ktorá ho ďalej vyvíjala pod názvom Apache Cordova. (Panhale, 2016)

Dnes je Adobe PhoneGap distribúciou Apache Cordovy alebo inými slovami, Cordova je motor poháňajúci PhoneGap podobne ako WebKit je motorom poháňajúcim Chrome a Safari (PhoneGap Blog, 2012). Obidva nástroje sú voľne stiahnuteľné z oficiálnych stránok a sú podporované známymi mobilnými operačnými systémami. PhoneGap okrem natívneho obaľovania poskytuje aj iné služby značky Adobe, ktorými Cordova nedisponuje. Medzi tie patrí napríklad PhoneGap Build a PhoneGap Hydration.

**PhoneGap Build** je cloudová služba, ktorá kompiluje kód aplikácie pre jednotlivé platformy (PhoneGap, 2016). K zostaveniu finálnej aplikácie teda nedochádza na lokálnom stroji vývojára, ale na cloude. Aplikáciu zabalenú do ZIP archívu stačí nahrať na server, ktorý sa postará o zvyšok. Hlavnými prínosmi tohto nástroja je uľahčenie údržby aplikácie, manažovanie tímovej práce na projekte a aktuálnosť verzií SDK[9] daných platforiem. Vývoj v cloude taktiež odbremení vývojára, ktorý nemusí mať nainštalované vývojové prostredie pre každú platformu.

**PhoneGap Hydration** rozosiela najnovšiu verziu aplikácie automaticky do zariadení vývojárov a testerov, ktorí si ju nemusia po každej kompilácii nanovo inštalovať. (PhoneGap, 2016)

Aktuálne PhoneGap podporuje 18 natívnych API volaní (Apache Cordova 21), medzi ktoré patrí napríklad prístup k akcelerometru, fotoaparátu, GPS, kontaktom, presunu súborov, informáciám o sieti, úložnému priestoru, upozorneniam, vibráciám. (PhoneGap, 2016)

### 2.6.2 Trigger.io

Trigger.io je ďalší nástroj, ktorý vytvára most medzi HTML, JavaScript kódom a natívnymi funkciami zariadenia.

Trigger.io je mladý komerčný projekt, ktorý pochádza z dielne firmy Webmynd. Pôvodne sa jednalo o nástroj na tvorbu rozšírení do webových prehliadačov, dnes však má Trigger.io pevné miesto v oblasti hybridného vývoja aplikácií. Aplikáciu vytvorenú použitím webových technológií, Trigger.io zabalí do svojho vlastného natívneho wrapperu. Ten sa v tomto prípade skladá z troch častí (Trigger.io, 2016):

1. natívne UI[10] komponenty,
2. natívne funkcie,

[9] Software Development Kit
[10] User Interface

3. most medzi JavaSriptovým kódom aplikácie a natívnymi funkciami zariadenia.

Aj keď je architektúra Trigger.io porovnateľná s architektúrou PhoneGapu, tak medzi týmito dvoma nástrojmi je možné nájsť zásadné rozdiely:

- Kompilácia aplikácie prebieha oveľa rýchlejšie ako u PhoneGapu. Na stránkach Trigger.io je uvedené, že sa jedná o pomer sekundy verzus minúty. Je to z toho dôvodu, že sa kompilácia rozdelí medzi lokálnymi nástrojmi a vzdialeným serverom. Celý proces zabezpečuje framework Forge, na ktorom je Trigger.io postavený. (Brady, 2012)
- Trigger.io má na platforme Android 5 krát rýchlejší natívny most – most medzi JavaScriptom a natívnym prostredím. (Dunn, 2012)
- Trigger.io nie je na rozdiel od PhoneGapu voľne použiteľný. Cena za používanie tohto nástroja sa pohybuje od 49 až po 1399 dolárov mesačne. (Trigger.io, 2016)

## 2.7 UI frameworky

Zatiaľ čo hybridné frameworky zaisťujú určitým spôsobom správny chod aplikácie, tak User Interface frameworky sa starajú o jej vzhľad. Pri tvorbe vzhľadu a užívateľského prostredia je častokrát časovo výhodnejšie použiť nejaký nástroj ako tvoriť vlastný dizajn. Práve preto sú čoraz populárnejšie UI frameworky, ktoré majú v sebe zabudované natívne prvky a komponenty kompatibilné s rôznymi platformami. Tie sú vytvorené väčšinou len pomocou CSS štýlov (poprípade JavaScriptu) a vďaka ním aplikácia nadobúda natívny rozmer.

Vývojár má k dispozícii na výber z pomerne veľkého množstva dostupných frameworkov. Po chvíli hľadania na internete je možné nájsť kvantum porovnávacích rebríčkov a odkazov, hlavne na framework Ionic, Sencha Touch, jQuery Mobile, Mobile Angular UI, Kendo UI Mobile, PhoneJS, Framework 7 a Famo.us.

V tabuľke 3 sú porovnané vlastnosti prvých piatich spomenutých UI frameworkov. Všetky tieto frameworky sú podporované troma najrozšírenejšími mobilnými operačnými systémami. Údaje vychádzajú z dokumentácií jednotlivých frameworkov a z aktuálnych štatistík získaných zo stránok Stackoverflow a Github. Úlohou tabuľky je uviesť v prehľadnej forme hlavné rozdiely, ktoré sú odrazovým mostíkom pre následný výber toho správneho UI frameworku.

Výber UI frameworku je dôležitou etapou pred samotným začiatkom vývoja aplikácie. Výber závisí na tom, čo vývojár chce dosiahnuť. Či aplikáciu, ktorá je určená len pre jednu platformu alebo pre všetky, či je požadované aby mala pôsobivé animácie alebo či je vývoj v tomto frameworku dostatočne podporovaný. Okrem voľne použiteľných nástrojov existujú aj komerčné alternatívy (Kendo UI). Tie na viac ponúkajú osobitý prístup a prémiové vývojové nástroje. Niektoré frameworky sú súčasťou rozsiahleho ekosystému (Sencha Touch, Kendo UI, Ionic), ktorý je väčšinou tvorený produktami a nástrojmi nie len pre samotný vývoj, ale aj pre manažovanie, analyzovanie, nasadenie a testovanie aplikácie. Ďalším dôležitým rozdielom je po-

Tabuľka 3: Porovnanie UI frameworkov

| | **Ionic** | **Sencha Touch** | **jQuery Mobile** | **Mobile Angular UI** | **Kendo UI Mobile** |
|---|---|---|---|---|---|
| Verzia | 1.2.4 | 2.4.2 | 1.4.5 | 1.2.1 | Q1 2016 SP1 |
| Rok vydania | 2013 | 2011 | 2010 | 2013 | 2011 |
| Cena | zadarmo | zadarmo / komerčná | zadarmo | zadarmo | skúšobná / 1000$ |
| UI komponenty | 20 vstavaných | 50 vstavaných | 27 vstavaných | 10 vstavaných + bootstrap | 14 vstavaných + vyše 70 jQuery komponent |
| Architektúra | MVW (AngularJS) | MVVM (ExtJS) | nie (jQuery) | MVW (AngularJS, Bootstrap) | MVVM (jQuery/integrácia s AngularJS) |
| Zakomponovanie do HTML | vlastné HTML tagy a atribúty | HTML v JavaScripte | HTML5 atribúty „data-*“ | vlastné HTML tagy a atribúty | HTML5 atribúty „data-*“ |
| Podpora | oficiálne fórum | oficiálne fórum, platená priama podpora | oficiálne fórum | oficiálne fórum | oficiálne fórum, platená priama podpora |
| Stackoverflow – počet výsledkov | 20 000 | 8 500 | 45 000 | 450 | 600 |
| Github rep. – počet hviezdičiek | 23 000 | - | 10 000 | 2 500 | - |
| Doplnky | Ionic Lab, Ionic Creator, Ionic View, Ionic CLI | grafová vizualizácia, Sencha Architect | Theme roller | Fastclick.js, Overthrow.js | Theme Builder, komplexný framework Kendo UI |

nuka UI komponent, ktorá je pre každý spomenutý framework špecifická. Sencha Touch a Kendo UI Mobile poskytujú vlastné grafické komponenty (nazývané často widgets), zatiaľ čo Ionic a jQuery Mobile len definujú prvky užívateľského rozhrania. Pri výbere UI frameworku je taktiež potrebné zvážiť jeho architektúru.

### 2.7.1 Konvencie medzi platformami

Rovnaký druh UI komponenty sa v rôznych operačných systémoch dizajnovo líši. Tento fakt je možné sledovať napríklad na tlačidlách, ktoré majú odlišný tvar, oblosť hrán, tieň, či veľkosť písmen.

Zatiaľ čo Apple od verzie iOS 7 prešiel k tzv. flatter dizajnovému štýlu, Google vsadil na materiálový dizajn. Zásadným rozdielom medzi zariadeniami s týmito systémami je absencia hardwarového spätného tlačidla v Apple zariadeniach. Z tohto

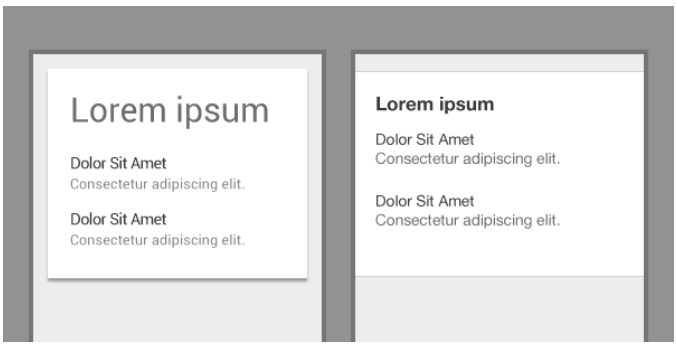
dôvodu musí byť v týchto zariadeniach užívateľské prostredie doplnené o UI prvok, ktorý umožňuje návrat na predchádzajúcu obrazovku. (O'Sullivan, 2014)

Výraznejšie konvencie je možné spozorovať hlavne na väčších komponentoch (blokoch) ako napríklad hlavička, navigácia, navigačná lišta, stavový riadok. Zatiaľ čo iOS centruje text v navigačnej lište, Android ho zarovnáva doľava. Podľa O'Sullivana (2014) je navigácia najväčším dizajnovým rozdielom medzi porovnávanými platformami. Tá sa v Androidoch nachádza v hornej časti obrazovky a častokrát býva schovaná v ľavom bočnom menu, ktoré sa vysunie po kliknutí na *hamburger* ikonu. IOS zas preferuje navigáciu prostredníctvom záložiek s ikonami, ktoré sú umiestnené v spodnej časti obrazovky.


Obrázok 4: Porovnanie navigácie v Android a iOS zariadení (O'Sullivan, 2014)

Ďalšou badateľnou odchýlkou je spôsob rozdeľovania jednotlivých obsahových častí. Každý obsah tvorí jednu tzv. kartu, ktorá pomáha užívateľovi v orientácii. Materiálový dizajn sa inšpiroval listom papiera, ktorému lemuje hrany tmavým tieňom. IOS vsadil na úplnú plochosť karty, ktorá sa rozprestiera cez celú šírku obrazovky.

Predvolený font v systéme Android je Roboto a na platforme iOS Helvetica Neue. Okrem fontu sa texty líšia aj veľkosťou, vzájomnou variáciou veľkosti textu (nadpis verzus obyčajný text) alebo množstvom voľného bieleho miesta medzi textami. (O'Sullivan, 2014)




Obrázok 5: Porovnanie fontov a kárt v Android a iOS zariadení (O'Sullivan, 2014)

Medzi týmito platformami okrem spomenutých rozdielov nájdeme rozdiely aj v štýle kontextovej navigácii, kontextových (upozorňovacích) okien, ikon alebo tlačidiel. Windows dlho bojoval s konštrukciou toho správneho a jednotného dizajnu z dôvodu existencie veľkého množstva verzií systému so svojim vlastným dizajnom. Najskôr začali tým, že definovali rozdielne dizajnové pravidlá pre telefóny, tablety a počítače. V súčasnosti dochádza k zjednoteniu dizajnu do sady pravidiel a pokynov UWP[11].

## 2.8 Backend

Lokálne úložisko dát je vhodnou voľbou pre aplikáciu, ktorú používa len jeden užívateľ a len na jednom zariadení. Aplikáciám, ktoré určitým spôsobom zdieľajú dáta medzi užívateľmi a zariadeniami, je na to potrebné pripraviť prostredie, často v angličtine nazývané backend. Pojem backend sa doslovne preloží ako zadná časť, no zvyčajne sa tento výraz neprekladá.

Jedná sa teda o tú časť aplikácie alebo webu, ktorá je bežným užívateľom skrytá a slúži na administráciu, k spracovaniu dát a zabezpečuje funkčnosť celej aplikácie. V redakčných systémoch backend umožňuje napríklad vkladať a upravovať články, v internetových obchodoch okrem zdieľania tovaru taktiež zakladanie užívateľských účtov alebo vyriaďovanie objednávok. (Adaptic, 2016)

Backend beží na serveri, ktorý môže byť vzdialený aj stovky kilometrov od zariadenia s aplikáciou. V klasických, serverovo-orientovaných, aplikáciách je práve backend to, na čom je celá aplikácia postavená. Dnes okrem klasického serverového programovacieho jazyka PHP, môže byť backend napísaný v kvante iných jazykoch ako napríklad Python, Ruby, Java, C#, vo frameworkoch od nich odvodených ako Django, Ruby on Rails a dokonca je možné na strane servera použiť JavaScript (napríklad Node.js, Express.js). Opak backendu je viditeľná časť aplikácie, frontend, a s ňou spojené operácie na klientskej strane.

[11]Universal Windows Platform

### 2.8.1 Backendové alternatívy

Morony (2016) vymedzuje tri backendové možnosti pre hybridnú mobilnú aplikáciu:

1. vytvorenie a hosťovanie vlastného backendu,
2. použitie softwaru,
3. použitie služby (Backend as a Service – BaaS).

#### Vytvorenie a hosťovanie vlastného backendu

Ako uvádza Morony (2016), jedná sa o DIY[12] prístup, pri ktorom je nutné manuálne nastaviť server, vytvoriť a následne spravovať databázu a naprogramovať prepojenie medzi aplikáciou a databázou. To všetko z hľadiska náročnosti zaberie spomedzi uvedených možností najviac práce a času.

Tento prístup umožňuje používať ľubovoľné technológie a vytvárať vlastné, na mieru šité, služby. Aplikácia a jej dáta sú kompletne vo vlastníctve vývojára, ktorý je zodpovedný za výpadky a údržbu servera. Ten je náročne škálovateľný[13] pri veľkom množstve užívateľov. (Morony, 2016; Stieben, 2012)

#### Použitie softwaru

Primárne sa jedná o software, ktorý zabezpečuje hlavne manipuláciu s dátami. V prípadoch, kedy nie je k dispozícii internetové pripojenie má za úlohu úlohou uskladniť dáta ľubovoľného formátu v lokálnej databáze a neskôr po pripojení ich automaticky synchronizovať so vzdialenou databázou na serveri. (Morony, 2016)

Príkladom takéhoto softwaru je Couchbase Mobile, ktorý je postavený na neštruktúrovanej Couchbase databáze. Couchbase má okrem klasickej serverovej časti aj mobilnú časť, ktorá sa nainštaluje do telefónu spoločne s aplikáciou. Synchronizáciu medzi jednotlivými časťami zabezpečuje nástroj Couchbase Sync Gateway. (TCL DigiTrade, 2016)

#### Použitie služby – Backend as a Service (BaaS)

Častokrát okrem skratky BaaS je možné natrafiť aj na skratku MBaas (Mobile Backend as a Service), ktorú je možné považovať za synonymum. BaaS predstavuje novodobý prístup k tvorbe mobilných a webových aplikácií a je presným opakom vytvorenia si vlastného serverového backendu. V tomto prístupe stačí k vytvorenému a funkčnému backendu napojiť aplikáciu, importovať model databázy a poprípade doplniť aplikáciu o služby tretích strán, ktorých integráciu backend podporuje.

Úlohou BaaS je jednoducho prepojiť mobilnú aplikáciu s cloudovou backendovou databázou a s API backendovej služby. Uľahčenie spočíva v odbremenení vývojára

[12] Do It Yourself

[13] Škálovateľnosť je žiadúca vlastnosť a schopnosť systému pracovať s náhlymi zmenami, vydržať náhlu záťaž a správne hospodáriť s výkonom. (Wikipedia, 2013)

od backendového spracovania dát, prepojenia aplikácie so serverom a od nutnosti jeho údržby. Namiesto tradičného servera sa pracuje na cloude, ktorý ponúka vývojárom množstvo ľahko použiteľných služieb tretích strán (ICT Revue, 2015). Najväčšími mínusmi tohto prístupu je podľa Moronyho (2016) značná závislosť aplikácie na týchto službách a cena za používanie. Tá sa zväčšuje úmerne k počtu užívateľom. Morony (2016) nakoniec upozorňuje na riziko ukončenia služby, s ktorým je spojená možnosť straty dát alebo nutnosť neľahkého prenosu kódu na inú backendovú platformu.

K BaaS je možné pristupovať z viacerých aplikácií, a teda nie je nutné vytvárať oddelený backend pre každú jednu. Tým dochádza k eliminácii často používaného kódu a všetko sa odohráva na jedinom mieste v rámci jedného modelu (Riggins, 2011). Prístup k backendovým službám je realizovaný cez špecifické REST[14] rozhranie, ktoré má jednotný tvar a je ľahko použiteľné.

BaaS poskytovatelia okrem backendu a úložiska dát ponúkajú sadu často potrebných funkcií, ktoré môže vývojár ľahko integrovať do aplikácie. Medzi tie patrí najčastejšie autentifikácia, správa užívateľov, emailové služby, integrácia sociálnych sietí, push notifikácie, analytické nástroje, integrácia platobných systémov, funkcie spojené s lokalizáciou a geografickou polohou, komunikácia v reálnom čase, offline funkčnosť.

Najznámejším a najpoužívanejším poskytovateľom BaaS bol Parse od Facebooku, ktorý však v januári 2016 vyhlásil koniec, čo značne ovplyvnilo mienku o backendových službách. Odvtedy sa čoraz viac hovorí hlavne o Firebase, Back&, Telerik, Kinvey, GameSparks.

## 2.9 NoSQL

NoSQL (Not only SQL alebo No SQL) databázy používajú na uskladnenie dát iné spôsoby ako tabuľkové modely, ktoré sú známe z tradičných relačných databáz. Termín NoSQL je známy od roku 1998 a nejedná sa len o garážový projekt, ale naopak, tento koncept bol dlhé roky cielene vyvíjaný po boku veľkých firiem ako Google, Amazon a Facebook. Dnes sa do tejto kategórii databáz zaraďujú stovky odrôd databázových produktov, čo zásadne rozširuje význam tohto pojmu. Jediné v čom sa tieto databázy zhodujú je nerelačný motív. Preto sa pojem NoSQL môže kľudne nahradzovať pojmami ako No RDBMS[15], No relational alebo NonRel. Rapídny rast počtu nerelačných databáz sa odôvodňuje nedostatkami relačných databázových systémov, ktoré prestávajú stačiť moderným webovým aplikáciám. (Fowler, 2015)

NoSQL systémy ponúkajú rýchly zápis a čítanie dát, a preto sú vhodnou voľbou pre Big Data aplikácie. NoSQL databázy nepotrebujú, a častokrát ani nemajú

[14]Representational State Transfer je architektúra, ktorá umožňuje vytvárať (create), čítať (read), meniť (update) a mazať (delete) dáta na serveri pomocou HTTP volaní. Súhrnne sa tieto operácie nazývajú CRUD. (Panhale, 2016)

[15]Relational DataBase Management System

schému, čo odbremení vývojárov od povinnosti dôkladného návrhu databázy. Prepojenia (väzby) medzi tabuľkami známe z RDBMS zanikajú a potrebné informácie sú ukladané ako jeden dlhý záznam. Tento záznam môže byť napríklad v JSON alebo XML formáte. Zatiaľ čo v relačných databázach je cieľom dosiahnuť normalizáciu dát, v NoSQL sú bežnosťou denormalizované duplikované dáta (Fowler, 2015)

NoSQL podporuje distribuovanú architektúru, a teda je možné rozdeliť jednu databázu medzi viacero strojov. Výhody plynúce z tejto vlastnosti sa využijú v aplikáciách, v ktorých sa musí spracovať také veľké množstvo dát (napríklad Facebook, Twitter správy), ktoré by nedokázal spracovať jeden, aj keď možno najvýkonnejší, server (Fowler, 2015). Dôležité je taktiež zmieniť, že opisované systémy sú zvyčajne voľne šíriteľné produkty.

Základom NoSQL je teda nerelačný dátový model, ktorý môže mať formu grafu, dokumentu, môže byť zoskupovaný do tzv. rodiny stĺpcov (column-family) alebo je usporiadaný do podoby kľúč-hodnota (key-value). Tabuľka 4 sumarizuje hlavné rozdiely medzi relačnými a nerelačnými databázami. (Fowler, 2015)

Tabuľka 4: Porovnanie vlastností SQL a NoSQL

| **Vlastnosť** | **SQL** | **NoSQL** |
|---|---|---|
| Organizácia dát | tabuľky | kľúč-hodnota, dokument, graf atď |
| Schéma databázy | predom navrhnutá | dynamicky vytváraná |
| Škálovateľnosť | vertikálna | horizontálna |
| Vlastnosti databázových transakcií | ACID | BASE |
| Vzťahy medzi dátami | cudzie kľúče, normalizácia dát | vnorené dokumenty, redundancia |
| Prístup k dátam | štandardizovaný jazyk SQL | volanie API metód, častokrát čiastočná podpora SQL jazyka |
| Spájanie dát | spojenie tabuliek – join | definovanie filtra a jeho hĺbky |
| Logika | procedúry, funkcie, spúšťače | obsiahnutá v kóde |

Nerelačné spracovanie je moderné a cloudové riešenie, ktoré je vhodné pre špecifické prípady. Zväčša ide o situácie vyžadujúce rýchlosť, dostupnosť, operácie v reálnom čase a prácu s veľkým objemom dát, ktoré sú zároveň nejakým (dokumentovým, grafickým, stĺpcovým) spôsobom orientované. Pre NoSQL systémy sú typické BASE garancie sľubujúce stálu dostupnosť (Basically Available), čiastočne a krátkodobo nekonzistentný stav (Soft-state), ktorý bude nakoniec po nejakom čase opäť konzistentný (Eventual consistency) (Fowler, 2015).

# 3 Metodika

Po zvládnutí zmienených teoretických znalostí je možné sa sústrediť na vývoj aplikácie. Pred zahájením samotnej implementácie je potrebné vykonať ešte pár nevyhnutných krokov. Dôležité je identifikovať a analyzovať funkčné požiadavky kladené na aplikáciu. Následne je vhodné zistiť záujem potenciálnych užívateľov o tieto funkcie. To je možné docieliť jednoducho pomocou dotazníkového šetrenia, ktoré poskytne určitú spätnú väzbu, objaví prehliadnuté nedostatky a prípadne prinesie ďalšie nápady a podnety.

Paralelne, počas priebehu šetrenia, je možné zvoliť vhodný databázový systém. Táto voľba je úzko spätá a odvíjajúca sa od zvolenej backendovej možnosti. Preto musia tieto voľby prebehnúť súčasne. Nasledujúce kroky vedú k výberu vhodného programovacieho jazyka a šikovného UI frameworku. Posledným krokom je voľba vývojárskych a testovacích nástrojov, ktoré umožňujú a urýchľujú vývoj.

## 3.1 Popis aplikácie

Aplikácia sa povahou zaraďuje do oblasti dopravy, avšak nie je určená len pre vodičov, ale aj pre bežných účastníkov cestnej premávky. Jej hlavnou úlohou je upozorňovať majiteľov vozidiel na nečakané a hlavne nežiadané situácie, ktorým môže vozidlo čeliť. Jej prvotný názov znel Carfriend.

Iniciátorom upozornenia je náhodný užívateľ aplikácie, ktorý je svedkom niektorej z nasledujúcich situácií: odťahovanie vozidla, kontrola parkovacích lístkov, rozdávanie papúč na kolesá, nehoda na parkovisku, vandalizmus, krádež a ďalšie. Ten o dianí upozorní majiteľa zaregistrovaného vozidla cez jeho evidenčné číslo. Majiteľovi príde upozornenie do aplikácie alebo na email a môže okamžite konať.

Okrem posielania upozornení konkrétnemu užívateľovi boli vymyslené ďalšie (prémiové) funkcie, ktoré by boli vhodné pre túto aplikáciu:

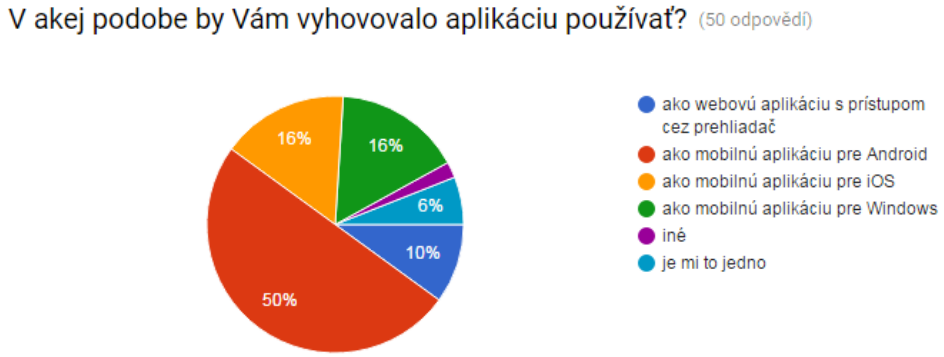
1. nebezpečná ulica – upozorňovanie na parkovanie na ulici, kde často dochádza k pokutám a nehodám,
2. predpoveď pokuty – aplikácia upozorní majiteľa vozidla na premnožené pokuty v oblasti, kde zaparkoval svoje vozidlo,
3. kontrola ulice – užívateľ jedným kliknutím zistí, či na aktuálnej ulici neprebieha blokové čistenie alebo iné parkovacie obmedzenie,
4. upozornenie na končiacu TK a EK vozidla.

Myšlienka aplikácie je postavená hlavne na ochote a spolupatričnosti účastníkov cestnej premávky pretože práve oni sú tvorcovia upozornení, a teda aj obsahu. Z toho dôvodu bolo medzi nimi uskutočnené dotazníkové šetrenie, ktorého úlohou bolo zistiť vlastnosti aplikácie tohto formátu. Pred samotnou sumarizáciou požiadaviek by som ozrejmil, že obdobná aplikácia v dobe začatia plánovania vývoja (rok

2015) minimálne na českom a slovenskom trhu neexistovala a nenašiel som podobnú aplikáciu ani v čase písania tejto práce.

Nasledujúce body sumarizujú hlavné výsledky dotazníkového šetrenia o vzorke 60 náhodných respondentov, pričom celý dotazník spolu s grafovým zobrazením výsledkov je vložený do príloh tejto práce.

A. Pri otázke, v akej podobe by respondentom najviac vyhovovalo aplikáciu používať, očakávane nedošlo k výraznej zhode.



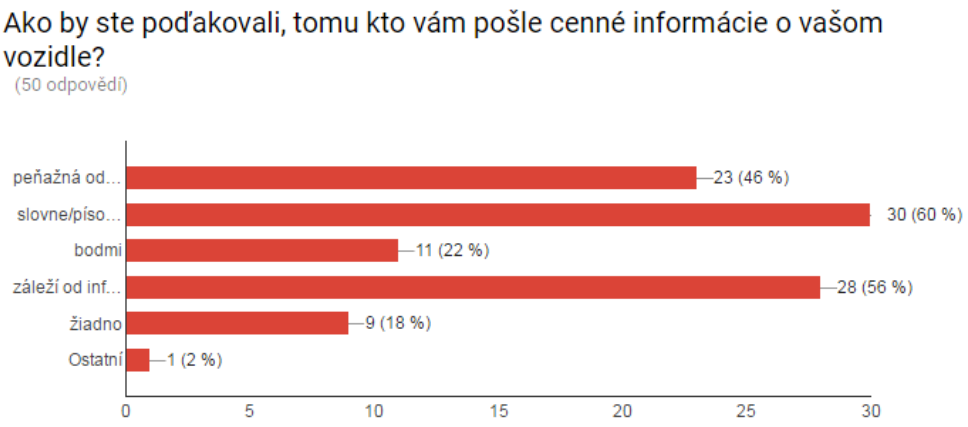

Obrázok 6: Otázka č. 5 z dotazníka

B. Najviac by motivovalo opýtaných k posielaniu oznámení v poradí peňažná odmena, vykonanie dobrého skutku, pocit karmy (že sa im to môže niekedy raz vrátiť) a závažnosť danej situácie (očitý svedok).

C. Vyše 80 % respondentov by bolo ochotných poslať denne aj viacej upozornení, ale k tomu by ich motivovalo v poradí najmä pozitívna skúsenosť s prijatím odmeny od niekoho, prémiové funkcie, zisk a jednoduchosť vytvorenia upozornenia.

D. Takmer každý opýtaný, ktorý vlastní vozidlo, by po prijatí upozornenia zareagoval podľa dôvernosti užívateľa, ktorý mu upozornenie poslal. Najviac opýtaných by ako prvé nadviazalo chat s danou osobou s cieľom zistiť viac informácií alebo by šlo situáciu ihneď skontrolovať.

E. Pri otázke, ako by majiteľ vozidla ocenil pravdivú informáciu, vyšlo najavo, že väčšina opýtaných by sa zariadila podľa povahy danej informácie. Viac respondentov by poďakovalo radšej písomne ako peňažne.




Obrázok 7: Otázka č. 24 z dotazníka

F. 75 % opýtaných si myslí, že prémiové funkcie by mohli byť užitočné pre vodičov a zhruba 80 % by aspoň raz použilo aplikáciu na zblíženie a zoznámenie s inými dopravnými účastníkmi.

Z výsledkov boli následne zostavené požiadavky na aplikáciu. Prvá požiadavka vychádza z bodu A., ktorý definuje multiplatformovú vlastnosť aplikácie. Namiesto vývoja verzie aplikácie pre každý jeden OS sa vývoj ubral hybridným webovým smerom. Druhá požiadavka, ktorá vzišla z bodov B., C. a E. je integrovanie platobného systému. Navyše bod C. zdôraznil, že je dôležité aby tvorba upozornenia bola užívateľsky prívetivá, krátka a jednoduchá. Podľa bodu D. by malo byť možné v aplikácii hodnotiť pravdivosť upozornení a na základe toho merať dôvernosť užívateľa. Bod D. spolu s bodom F. definovali požiadavku na integrovanie internetovej komunikácie (chat). V bodoch F. a C. opýtaní požiadali o zahrnutie dodatočných (prémiových) funkcií určených pre vodičov. Bod F. taktiež čiastočne naznačuje, aby aplikácia mala sociálne prvky (napríklad vytváranie priateľstiev, posielanie poznámok medzi užívateľmi) na zbližovanie sa s ostatnými účastníkmi cestnej premávky.

### 3.1.1 Analýza požiadaviek

Každá aplikácia by mala zohľadňovať čo najväčší počet požiadaviek užívateľov avšak nie je možné akceptovať a uspokojiť všetky. Dôvodmi môže byť legislatíva, implementácia, nevhodnosť požiadavky alebo podobnosť s konkurenčným produktom. Preto je vhodné najskôr požiadavky zanalyzovať a porovnať s prípadnými konkurenčnými službami.

#### Kontrola blokového čistenia

Aby bola táto funkcia dostupná napríklad pre mesto Brno, je potrebné získať informácie o blokových čisteniach v meste. Nakoľko je každé čistenie organizované samosprávou danej mestskej časti, tak nie je k dispozícii kompletný zoznam čistení

v celom Brne. Z toho dôvodu sa dáta musia získať manuálne z webových stránok jednotlivých častí mesta. Každá mestská časť navyše uverejňuje tieto dáta v rôznych formátoch, od klasického pdf súboru až po obrázok. Blokové čistenia prebiehajú v každej časti v rôzny čas a zároveň si samospráva vyhradzuje právo zmien, doplnenia alebo vynechania termínu blokového čistenia. Preto je udržanie konzistencie dát takmer nemožné.

Na internete je možné nájsť aplikáciu, ktorá spolupracuje s 8 brnianskymi samosprávami. Aplikácia sa nachádza na stránkach `http://cisteniulic.cz/` a okrem Brna pokrýva mestá Tábor, Pardubice, Ústí nad Orlicí, Třebíč, Uherské Hradiště a Zlín. Po registrácii aplikácia požiada užívateľa aby zvolil ulicu, ktorú chce sledovať. Keď sa blíži blokové čistenie na tejto ulici, aplikácia upozorní užívateľa pomocou SMS alebo emailu. Výhodou tejto aplikácie je priama podpora a odporúčanie od jednotlivých samospráv a možnosť prehliadania blokových čistení na mape. Nevýhodou je spoplatnenie tejto služby a chýbajúca možnosť tvorby dotazov v reálnych situáciách. Ročné sledovanie jednej ulice (behom roka môže prebehnúť čistenie na ulici 1 až 4 krát) s SMS upomienkou stojí 20 Kč, štyroch ulíc 50 Kč a dvanásť ulíc je možné sledovať za 100 Kč.

Z recenzie aplikácie cisteniulic.cz uverejnenej na `http://www.idnes.cz/` a z náležitej verejnej diskusie vyplýva, že o túto funkciu majú záujem nie len starostovia samospráv ale aj občania. Väčšina redakciou opýtaných vodičov si ju je ochotná zaplatiť. (Horáková, 2015)

Informácie o blokových čisteniach sa budú ukladať do databázy v nasledujúcom formáte:

```
[{"locality":"Brno", "sublocality":"Brno-střed", "route":"Cihlářská",
"country":"Česká republika", "from":"8:00:00", "to":"11:00:00",
"date":"06/14/2016", "info":"+parkovisko"}]
```

### Upozorňovanie na končiacu TK a EK vozidla

Podľa zákonodarca (predpis 302/2001 Sb.) je povinnosťou majiteľa vozidla vykonať technickú a emisnú kontrolu v lehotách stanovených zákonom. Technická kontrola (TK) zisťuje funkčnosť technických súčastí vozidla nevyhnutných na jeho správnu prevádzku, emisná kontrola (EK) preveruje, či nedochádza k vypúšťaniu nadnormálneho množstvo škodlivín do ovzdušia a obidve kontroly vykonáva stanica technickej kontroly (STK).

Lehota je určená typom vozidla, no prevažne vždy je jej dĺžka meraná v rokoch. Nakoľko sa jedná o dlhé obdobie, častým predmetom pokuty je práve zabudnuté vykonanie jednej z týchto kontrol. Takáto pokuta môže skončiť až zabavením technického preukazu. Od 1. marca 2016 sa dokonca spustil na Slovensku systém, ktorý každý deň vyhľadáva v databáze vozidlá s neplatnou technickou alebo emisnou kontrolou a následne ich majiteľom automaticky posiela pokuty. Výška pokuty za každú z dvojice neplatných kontrol je 165 €, čo v súčte znamená 330 €. Zároveň v rámci projektu JISCD (Jednotný Informačný Systém v Cestnej Doprave) bola na

Slovensku spustená služba, ktorá včas informuje majiteľa vozidla o končiacej platnosti kontroly. Majiteľ o túto službu musí najskôr požiadať, a to buď vykonaním registrácie každého vozidla alebo založením elektronického občianskeho preukazu a prihlásením sa cez čítačku kontaktných čipových kariet. O tejto službe väčšina občanov nevie.

České a ani slovenské ministerstvo dopravy neposkytuje žiadne API na prístup k termínom platnosti kontrol vozidiel. Z toho dôvodu bude možné implementovať funkciu spôsobom, ktorý bude očakávať od užívateľa vstupné dáta o týchto termínoch a v určenom predstihu užívateľa informuje.

#### Platobný systém

Najvhodnejší platobný systém pre túto aplikáciu by mal prebiehať v reálnom čase. Platby by tiekli priamo od vodičov k užívateľom a aplikácia by si vždy strhla určité percento. Predpokladá sa, že užívatelia by najradšej platili kreditnými resp. debetnými kartami z dôvodu ich pohodlného používania. Tieto charakteristiky smerujú k zavedeniu tzv. P2P platieb. Peer-to-Peer alebo Person-to-Person platby prebiehajú priamo medzi dvoma osobami, sú veľmi jednoduché, rýchle a na ich uskutočnenie stačí mobilný telefón. P2P platby, novodobý trend v bankovníctve, zavádzajú veľké internetové aplikácie (Facebook, Google), platobné inštitúcie (PayPal, Venmo, Braintreepayments), obchody a čoraz intenzívnejšie aj české a slovenské banky (aplikácie VIAMO, Friends24).

Aby užívateľ mohol realizovať P2P platby, musí mať založený účet v danej inštitúcii a platiť cez jej aplikáciu. Tento variant nie je vhodný pre vyvíjanú aplikáciu, nakoľko by nútil užívateľa k inštalácii ďalších aplikácií. Vhodnejšie je použiť API nejakého platobného systému. Tu sa však častokrát naráža na nedostupnosť služieb. Z tohto dôvodu nie je možné použiť ani napríklad ostrieľaný systém Stripe.

Do úvahy spadá už len PayPal a Braintreepayments. Tieto systémy zároveň ponúkajú službu, ktorá dokáže rozdeliť jednu čiastku medzi viaceré osoby. To je pre túto aplikáciu hľadaná alternatíva P2P platieb.

## 3.2 Voľba databázového systému

Dáta, ktoré aplikácia bude využívať je potrebné niekde uskladniť. A to platí ako aj pre dáta vytvorené užívateľmi (správy, upozornenia), tak aj pre tie, ktoré boli vložené z administratívnych dôvodov. Výber databázového systému je úzko spätý s backendovým riešením. Preto je potrebné najskôr zvoliť vhodné backendové riešenie.

### 3.2.1 Výber backendovej alternatívy

V predchádzajúcich kapitolách boli zmienené a opísané tri backendové možnosti. V tejto časti bude jedna konkrétna možnosť zvolená. Nakoľko sa bude aplikácia zaradzovať do jednostránkových webových aplikácií, a teda bude napísaná len vo

frontendových jazykoch, tak nie sú pre ňu potrebné žiadne serverové akcie. Server bude len zdrojom dát, ktoré poskytne cez REST rozhranie. Hotové REST rozhranie a databázu ponúka backendová služba. Tá zároveň ponúka možnosti integrácie služieb tretích strán, ktoré sú takisto potrebné v tejto aplikácii. Konkrétne komunikáciu v reálnom čase, autentizáciu, funkcie s geografickou polohou a push notifikácie. Použitie BaaS je zároveň novodobým trendom a predmetom odporúčaní v mnohých internetových diskusiách. To sú dôvody, prečo bol nakoniec zvolený BaaS poskytovateľ.

Po porovnaní jednotlivých backendových poskytovateľov figurujúcich na momentálnom trhu bol vybratý veľmi mladý poskytovateľ so syntakticky podobným názvom Backand (Back&). Na jeho stránkach[16] sa uvádza, že je to mocný BaaS pre AngularJS a ReactJS poskytujúci Ionic integráciu, komunikáciu v reálnom čase a množstvo iného. Keďže sa jedná o pomerne mladý a neustále vyvíjajúci sa projekt, tak do mája 2016 je kompletne zadarmo. Následne sa ceny vyrovnajú ostatným poskytovateľom a bezplatný účet bude obsahovať 30 žiadostí behom sekundy, 150 pripojení v reálnom čase, 20 GB úložného priestoru a 2 GB priestoru pre databázu. Tento účet je dostačujúci pre prvotné nasadenie a testovanie aplikácie. V prípade potreby je ho možné za 1 300 dolárov rozšíriť až na 160 žiadostí za sekundu a 150 000 pripojení v reálnom čase.

### 3.2.2 Popis databázy

Nakoľko bola zvolená na backend služba, tak je nutné akceptovať databázový systém, ktorý služba ponúka. Back& poskytovateľ ponúka na modelovanie databázy NoSQL rozhranie, ktoré nadefinuje dátový model v JSON formáte. Nová databáza vytvorená v Backand rozhraní sa v skutočnosti ukladá do relačných databáz Amazon Web Services. Backand následne okolo relačnej databázy vytvorí NoSQL prostredie. Z toho dôvodu je možné vyexportovať ER diagram databázy a zároveň sa na ňu dotazovať NoSQL spôsobom. Výsledkom dotazu môže byť cudzí kľúč (jednoduché ID) alebo záznam so všetkými vnorenými objektami. Zmenou jedného parametru v API volaniach je možné tento typ výsledku ľahko meniť. Taktiež je možné pri volaní nastaviť do akej hĺbky má dotaz pôsobiť (parametre `deep`, `level`), či je potrebné vrátiť súvisiace objekty (`relatedObjects`), výsledky nejakým spôsobom zoradiť (`sort`), filtrovať (`filter`) alebo stránkovať (`pageSize`, `pageNumber`).

Veľkou výhodou je možnosť importu existujúcej MySQL, PostgreSQL, MS SQL alebo ORACLE databázy, ktorej štruktúru Backand zanalyzuje a vytvorí nad ňou svoje vlastné rozhranie. Takúto možnosť neponúka väčšina popredných BaaS poskytovateľov.

[16] `Backand.com`

## 3.3 Zvolený programovací jazyk

Programovací jazyk mení nápady a myšlienky na reálne prevedenie. Umožňuje vytvárať algoritmy, generovať užívateľské rozhranie a zamestnávať databázový systém. Preto jeho voľba patrí medzi najzodpovednejšie úlohy pri plánovaní vývoja aplikácie. Od vlastností programovacieho jazyka sa následne odzrkadľujú slabé a silné stránky aplikácie.

Vzhľadom na to, že implementovaná aplikácia je webová a má prívlastok jednostránková, pripadá do úvahy programovací jazyk JavaScript, ktorý bude súbežne dopĺňaný technológiami HTML a CSS. HTML vytvára štruktúru webovej stránky a CSS jej dodáva vzhľad a formát. Tvorba komplexnej aplikácie len v čistom JavaScripte je síce možná ale nie je moc preferovaná. Namiesto toho sa odporúča použiť JavaScriptový framework, ktorý výrazne uľahčuje prácu.

Zvolený bol framework AngularJS, pretože ponúka stabilitu, ktorá poháňa tisíce aplikácií. Tento framework je zároveň ešte stále považovaný za najobľúbenejší a najrozšírenejší JavaScriptový framework. AngularJS je voľne dostupný pod MIT licenciou, ktorá ho umožňuje používať na súkromné a aj komerčné účely. Založený bol spoločnosťou Google, ktorá ho predstavila v roku 2012. Medzi najdôležitejšie koncepty tohto frameworku zaraďuje Mrozek (2012) Two Way Data-Binding, implementáciu Dependency Injection, direktívy, klientske šablóny a znovu použiteľné komponenty.

MVC[17] architektúra jasne oddeľuje zobrazovaciu logiku od aplikačnej, ktorá je obsiahnutá v controlleroch. Controller je JavaScriptový objekt, ktorý má kontrolu nad dátami a viaže sa k danej šablóne špecifikáciou atribútu `ng-controller` v rodičovskom HTML elemente. Každý controller má svoj vlastný `scope` slúžiaci na uskladnenie objektov a premenných, ktoré sú na ňom definované.

Aktuálne sa pracuje na úplne novej verzii AngularJS, ktorá prinesie zásadné zmeny oproti predchádzajúcim verziám.

Pri programovaní v AngularJS je vhodné dodržiavať predom zvolený štýl pre syntax, konvencie a štruktúru aplikácie. Použitím štýlu sa vytvára optimalizovaný, prehľadný a čistý kód. Na výber je množstvo príručiek od rôznych programátorov, zvolená bola príručka od autora Johna Papaa. Tá okrem iného definuje nasledovné pravidlá (Papa, 2016):

- 1 komponent v jednom súbore a menej než 400 riadkov kódu,
- nedeklarovať AngularJS modul do premennej,
- používať `controllerAs` syntax, čím sa eliminuje používanie `$scope`,
- používať konštantnú premennú, ktorá odkazuje na `this`,
- umiestniť priradenie metód na vrch controllera,
- používať radšej `factory` ako `service`.

[17] Model-View-Controller

## 3.4 Voľba UI frameworku

Spomedzi porovnávaných UI frameworkov bol pre túto aplikáciu zvolený Ionic framework. Tento framework ponúka bohaté množstvo komponentov, definuje jasnú štruktúru aplikácie, je vynikajúco zdokumentovaný a je ho možné používať pod MIT licenciou. Jedná sa o pomerne mladý, ale rapídne rozrastajúci sa projekt s aktívnou komunitou a prakticky využiteľnými doplnkami. Ako uvádza Josh Morony (2015) vo svojom článku *8 dôvodov prečo som rád, že som prešiel na Ionic framework*, je bezpochyby jasné, že Ionic je dominantným HTML5 mobilným frameworkom v súčasnosti a má ambície byť v budúcnosti ešte lepším.

Ionic je postavený na ekosystéme obsahujúcom webový framework AngularJS a natívny zapuzdrovač Apache Cordova. Vizuálnu stránku aplikácie dotvára Ionic pomocou kaskádových štýlov CSS, ktoré spoločne budujú jednoduché, prispôsobiteľné a natívne užívateľské prostredie. Ionic prináša do HTML štruktúry svoje vlastné predpripravené HTML elementy. Tieto elementy sa nazývajú aj smernice (direktívy) a je ich možné ľubovoľne rozširovať.

Ionic je aktuálne v stabilnej verzii 1.2.4 (Copenhagen), no však paralelne sa vyvíja a je k dispozícii skúšobná verzia Ionicu (v. 2.0 beta) s AngularJS 2. Obidve verzie ponúkajú rovnaké komponenty líšiace sa prevažne syntaxou, ktorú udáva nová verzia jazyka AngularJS, najnovšia verzia JavaScriptu ES6 a rozšírenie nad JavaScriptom TypeScript.

Medzi ďalšie vlastnosti a výhody tohto UI frameworku patrí:

- pre efektívnu manipuláciu s CSS vlastnosťami Ionic ponúka preprocesor SASS[18],
- Ionic aplikáciu je bezproblémové spustiť v súčasnosti na Androide 4.1 alebo vyššom, iOS 7 alebo vyššom a na Windows 10 (Drifty Co, 2016),
- Ionic disponuje voľne použiteľným rozsiahlym balíčkom ikon pre každý podporovaný operačný systém,
- na sprístupnenie natívnych funkcií môže Ionic integrovať rozšírenie ngCordova. NgCordova je kolekcia AngularJS služieb a rozšírení, ktorá bola vytvorená Ionicom na uľahčenie a urýchlenie práce s Cordovou. Zároveň ngCordova umožňuje používať promises[19], ktoré upracú kód a pomôžu vyhnúť sa problémom spojeným s asynchrónnym volaním (Wilken a Bradley, 2015).

Spoločnosť Ionic okrem samotného frameworku Ionicu poskytuje aj doplňujúce nástroje a služby. Najpoužívanejším nástrojom je Ionic príkazový riadok (CLI), prostredníctvom ktorého je možné vykonávať všetku Ionic mágiu. Na testovanie, náhľad a zdieľanie aplikácie je vynikajúca bezplatná aplikácia Ionic View, ktorá je dostupná na platformy iOS a Android. Pomocou služby Ionic Creator je možné

[18] Syntactically Awesome Style Sheets

[19] Promise je objekt reprezentujúci hodnotu, ktorá nemusí byť ešte dostupná, ale bude získaná v budúcnosti. Promise umožňuje napísať asynchrónny kód vo viac synchrónnom spôsobe. (Panda, 2013)

jednoducho navrhnúť vzhľad aplikácie pomocou techniky drag and drop. Svoje uplatnenie si nájdu aj nástroje Ionic Lab, Deploy, Analytics, Playground, Push a Market.

## 3.5 Použité vývojové nástroje

Keďže na vývoj webovej resp. hybridnej aplikácie je možné použiť v podstate akékoľvek prostredie, výber editoru bol založený na subjektívnych preferenciách autora. Preto bol zvolený voľne dostupný editor **NetBeans IDE**. Tento editor je napísaný v jazyku Java, beží na operačných systémoch s Java Virtual Machine a má integrovanú podporu pre JavaScript, HTML a CSS.

Ladenie a testovanie aplikácie prebiehalo prevažne vo vývojárskych nástrojoch prehliadača Google Chrome, Android emulátore a v reálnych mobilných zariadeniach.

Softvér **Visual Paradigm** bol využitý na tvorbu UML modelov. UML[20] je grafický modelovací jazyk, ktorý slúži na vytváranie vizuálnych modelov. Tento jazyk patrí do skupiny objektovo-orientovaných jazykov. UML popisuje 13 oficiálnych typov diagramov a medzi tie najhlavnejšie sa radí diagram tried, aktivít, časovania, komunikačný a stavový diagram a diagram prípadov užitia. Po analýze požiadaviek je vhodné graficky zobraziť funkcie aplikácie, ich vzájomné prepojenie a interakciu medzi aplikáciou a aktorom (užívateľom). Diagram, ktorý je na takúto vizualizáciu určený sa nazýva diagram prípadov užitia (Use Case diagram). Jeho podoba je zobrazená vo výsledkoch práce.

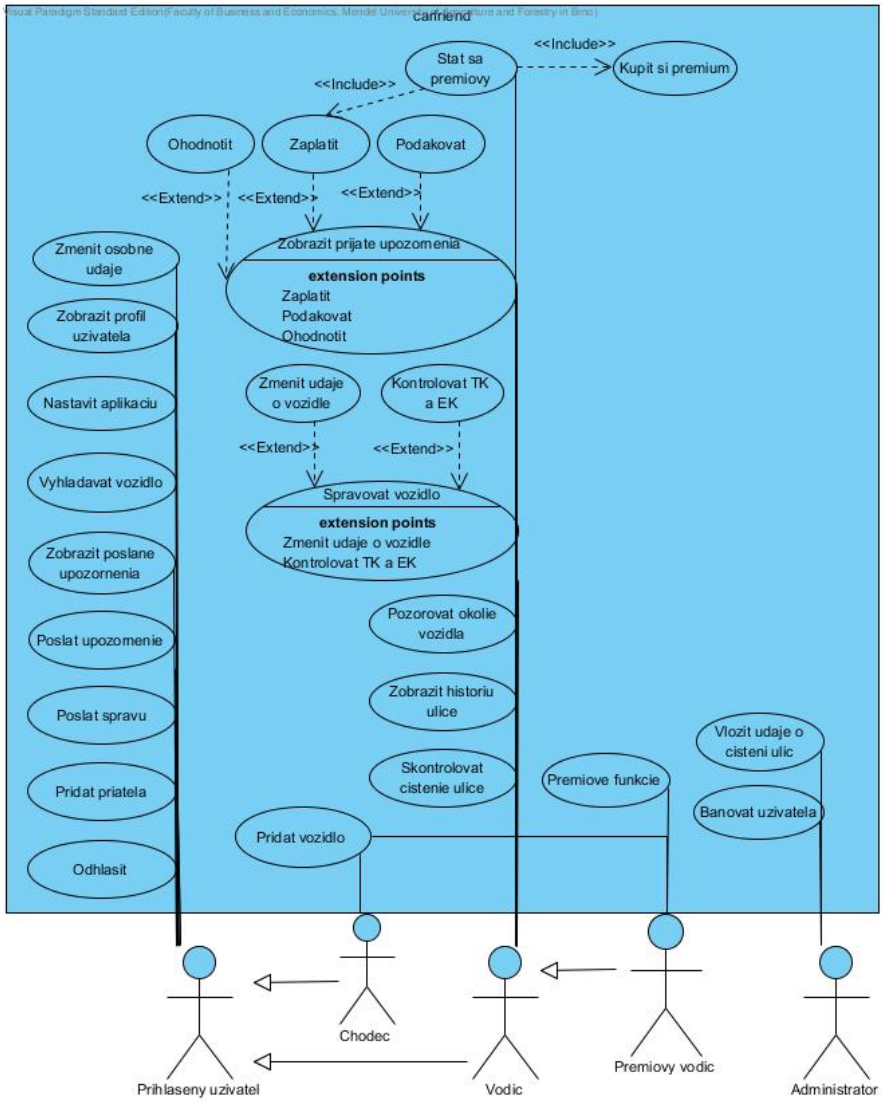
[20]Unified Modeling Language

# 4 Výsledky

## 4.1 Návrhy aplikácie

Pred začatím implementácie je dôležité vytvoriť návrhy aplikácie. Návrhy prebiehali v dvoch krokoch. V prvom kroku sa zostavil diagram prípadov použitia, ktorý predstavuje funkčnú časť aplikácie. Dátovú časť aplikácie popisuje entitno-relačný diagram. Obidva diagramy boli zhotovené na začiatku a s pribúdajúcou funkcionalitou boli priebežne dopĺňané.

### 4.1.1 Diagram prípadov užitia




Obrázok 8: Diagram prípadov užitia

Diagram prípadov užitia na obrázku 8 graficky znázorňuje hlavné funkcie systému. S aplikáciou pracujú traja aktori. Administrátor a prihlásený užívateľ, ktorý má dva varianty. Tie sa v use case diagrame značia pomocou generalizácie. Od povahy daného užívateľa následne plynú právomoci a možnosti využitia systému. Každý prihlásený užívateľ bude môcť napríklad poslať upozornenie inému užívateľovi, zobraziť históriu poslaných upozornení, vyhľadávať vozidlá a zobraziť ich profily. Všetky jeho možnosti sú uvedené v diagrame.

Aktor `Chodec` si môže pridať nové vozidlo, čím sa z neho automaticky stane vodič a otvoria sa mu nové možnosti. Iné funkcie chodec nemá nakoľko nemôže byť adresátom upozornenia a taktiež nemôže zabezpečiť neexistujúceho vozidlo.

Aktor `Vodic` môže iniciovať hneď niekoľko prípadov užitia. Napríklad môže spravovať svoje vozidlo, čo znamená, že môže zmeniť jeho údaje (okrem evidenčného čísla a farby) alebo môže pridať a kontrolovať termín technickej a emisnej kontroly. Ak vodič svoje vozidlo zaparkuje, tak ho môže v systéme na obmedzenú dobu (30 minút) zabezpečiť. Zabezpečenie zahŕňa kontrolu čistenia danej ulice, informácie o histórii nehôd a pokút na tejto ulici a hlásenie priestupkov v okolí. Po prijatí upozornenia môže upozornenie ohodnotiť, či bolo reálne alebo vymyslené, poďakovať zaň alebo odmeniť odosielateľa peniazmi.

Bežný vodič si môže pridať len jedno vozidlo, zatiaľ čo jeho prémiový variant si ich môže pridať hneď niekoľko. `Premiovy vodic` zároveň môže používať prémiové funkcie, ktoré sa líšia od bežných tým, že lehota a frekvencia ich používania je neobmedzená. Vodič získa prémiové možnosti, keď cez aplikáciu pošle v súčte určitú sumu peňazí alebo si priamo kúpi prémiový účet.

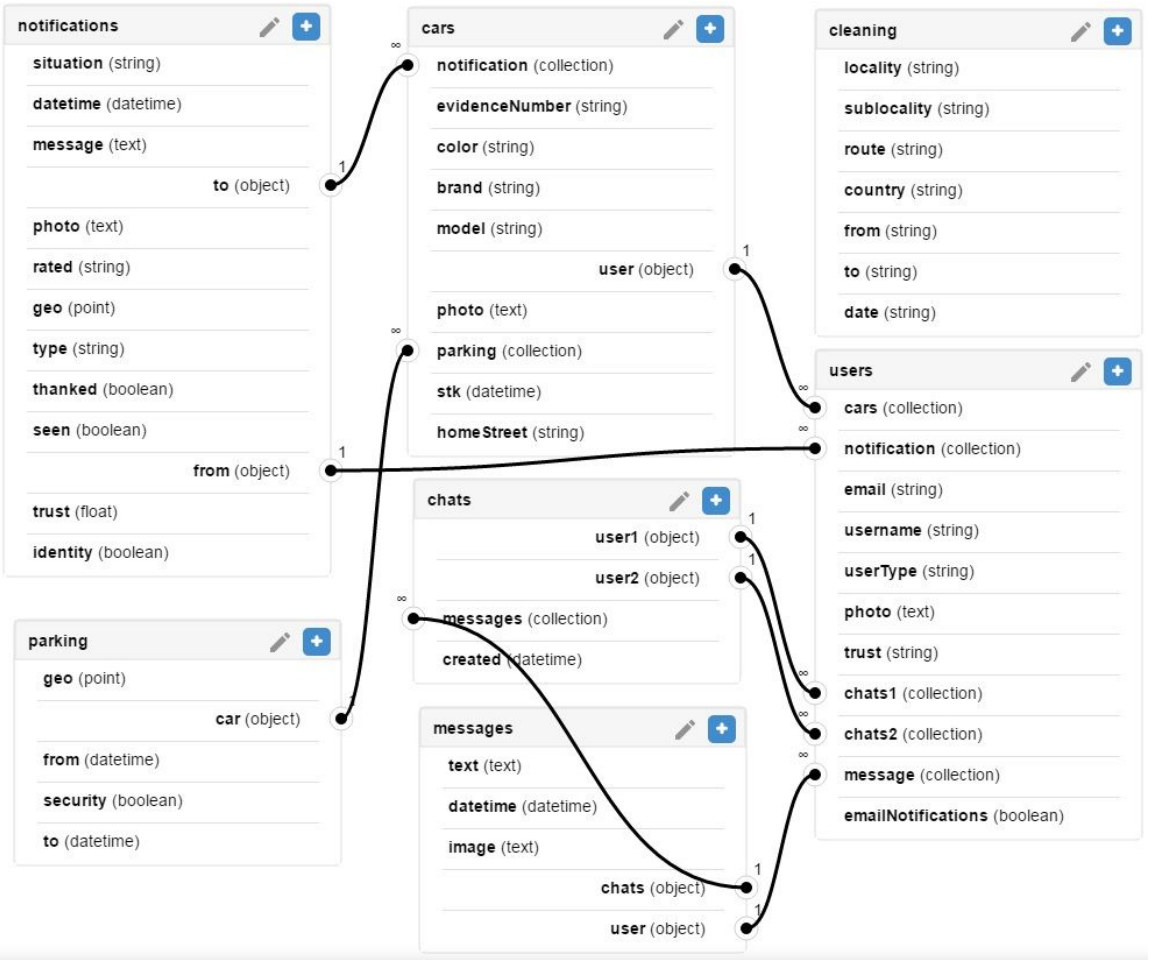
Aktor `Administrator` nemá v tomto momente skoro žiadne povinnosti. Jeho hlavnou úlohou je pridávať a aktualizovať informácie o blokových čisteniach. Takisto kontroluje užívateľov, ktorým môže v odôvodnených prípadoch zablokovať účet.

Predpokladom na to, aby užívateľ mohol používať aplikáciu je registrácia a následné prihlásenie do systému a pre administrátora nadobudnutie administratívnych práv.

### 4.1.2 Dátový model

Dátový model bol vytvorený v Back& rozhraní a na jeho modelovanie sa využil JSON formát. Ako už bolo spomenuté, toto rozhranie ponúka kúsok z NoSQL v relačnom SQL svete. Z toho dôvodu pri tvorbe modelu existujú obmedzenia z obidvoch svetov. Všetky názvy entít a atribútov v modeli, ako aj následné ukážky zdrojových kódov, budú uvedené v angličtine.

Rozhranie ponúka na výber 6 štandardných dátových typov, pričom väčšina použitých typov je nakoniec *string*. Zaujímavým typom je *point*, ktorý je priamo určený na uloženie GPS súradníc. Jedná sa o pole s dvoma hodnotami, napríklad [10,20]. Obrázok sa ukladá ako text vo formáte *base64*, ktorý zobrazuje binárne dáta pomocou ASCI znakov. Stopy NoSQL je možné sledovať napríklad na značnom množstve stĺpcov v jednotlivých tabuľkách.




Obrázok 9: Entitno-relačný diagram

Tabuľky sa spájajú pomocou dvojice kľúčov *object* a *collection*, pri čom sa vždy použije väzba 1:N. Spojenie tabuľky user a notification v JSON formáte má nasledujúcu podobu.

```
[
  {"name": "users",
    "fields": {
      "notification": {
        "collection": "notifications",
        "via": "from"
       }
    }
  },
  {"name": "notifications",
    "fields": {
      "from": {
```

```
        "object": "users"
      }
    }
  }
]
```

Upozornenie sa v databáze viaže na konkrétne vozidlo a nie na majiteľa. Predpokladá sa totižto, že bude existovať v aplikácii možnosť predaja vozidla, s ktorým by sa preniesla aj celá jeho história. To sa už netýka konverzácií, ktoré sa viažu na konkrétneho užívateľa. Dvaja užívatelia najskôr medzi sebou zahája chat, ktorý je následne plnený správami. Tento návrh umožňuje získavať jednoduchým dotazom všetky správy chatu.

Majiteľ vozidla bude v aplikácii vystupovať pod evidenčným číslom svojho vozidla, zatiaľ čo užívateľ bez auta bude vystupovať pod prezývkou. Tabuľka cleaning obsahuje manuálne importované dáta o blokových čisteniach. Stĺpce tabuľky sú pre pohodlnejšiu prácu pomenované rovnako ako kľúče v JSON dátach, ktoré vracajú Google mapy. Zároveň táto tabuľka ako jediná nemá žiadne závislosti. Dáta z tabuľky parking je možné využiť napríklad aj na zaznamenávanie histórie navštívených miest.

Je na mieste spomenúť, že dolovanie dát z databázy je možné predom obmedziť. V Back& na to slúžia tzv. preddefinované filtre, ktoré načítajú len tie dáta, ktoré sú spojené s aktuálnym užívateľským menom alebo užívateľkou rolou. V systéme sú na to vyhradené dve premenné `sys:username` a `sys:role`. Filtre sa využívajú pri výpise chatov užívateľa, jeho vozidiel alebo poslaných upozornení a je ich možné zostrojiť v NoSQL alebo SQL jazyku.

## 4.2 Príprava projektu

V tejto podkapitole je popísaný postup inštalácií jednotlivých prerekvizít potrebných pri implementácii aplikácie a taktiež aj spôsob vytvorenia nového projektu.

### 4.2.1 Prerekvizity

V prvom rade je potrebné nainštalovať na lokálnom počítači nástroj Node.js. Ten následne, okrem iného, sprístupní správcu balíčkov *npm*[21], pomocou ktorého je možné nasťahovať všetky potrebné knižnice a frameworky do lokálneho počítača. Po úspešnej inštalácii je možné z príkazového riadka spustením príkazu `npm install -g cordova ionic` nainštalovať najnovšie verzie prerekvizít Apache Cordova a Ionic SDK.

Pre testovanie aplikácie v emulátoroch alebo cez USB rozhranie je potrebné nainštalovať SDK daného operačného systému.

[21]Node Package Manager

### 4.2.2 Vytvorenie projektu

Nový Ionic projekt sa vytvorí pomocou príkazového riadka CLI, ktorý sa nainštaluje spolu s najnovšou verziou Ionicu. Spustením nasledujúceho príkazu sa vygeneruje v pracovnom adresári Ionic projekt s názvom carfriend, ktorý je založený na kartovej šablóne:

```
ionic start carfriend tabs
```

Okrem uvedenej šablóny je možné vytvoriť prázdny projekt (`blank`) alebo projekt s bočným menu (`sidemenu`).

### 4.2.3 Základné príkazy

- Nastavenie SASS sa vykoná spustením príkazu `ionic setup sass` v adresári s Ionic projektom.
- Aby bolo možné prehliadať aplikáciu cez Ionic View je potrebné mať založený účet na `ionic.io` a príkazom `ionic upload` nahrať aplikáciu na server.
- Pre spustenie aplikácie v lokálnom prehliadači existuje príkaz `ionic serve`, ktorý automaticky obnovuje stránku po zmene zdrojového kódu.
- Preklad projektu a spustenie na lokálnom emulátore pre *nazovPlatformy* sa vykoná sledom nasledujúcich príkazov: `ionic platform add` *`nazovPlatformy`*, `ionic build` *`nazovPlatformy`* a `ionic emulate` *`nazovPlatformy`*.
  *Pozn.: pre emulovanie na iOS simulátore je potrebný Mac.*

## 4.3 Implementácia

Po vytvorení projektu je možné zahájiť implementáciu. Tá prebieha za už zmienených podmienok. Počiatočná implementácia sa odohráva v troch súboroch `app.js`, `config.js` a `routes.js`. V prvom súbore sa deklarujú použité moduly resp. rozšírenia, druhý súbor slúži na konfiguráciu týchto modulov a v treťom súbore sa definujú stavy aplikácie.

### 4.3.1 Použité moduly

Pre dosiahnutie určitých funkcionalít boli použité voľno dostupné nasledujúce moduly (direktívy) a rozšírenia:

- Angular-xeditable – balík AngularJS direktív, ktoré umožňujú vytvárať editovateľné elementy,
- Angular filter – balík užitočných filtrov pre AngularJS šablóny,
- Ionic Material – knižnica s CSS a JS rozšíreniami, ktoré do Ionic aplikácie vnášajú nádych materiálového dizajnu,

- Ionic color picker – paleta farieb, ktorá podporuje výber konkrétnej farby,
- Ionic date picker – obdobne ako aj color picker, len na výber dátumov,
- Ionic close popup – direktíva, ktorá umožňuje zatvoriť Ionic popup okno kliknutím hocikam na obrazovku,
- Angular SVG round progressbar – AngularJS modul, ktorý využíva SVG na vytvorenie kruhového ukazovateľa priebehu,
- ngCordova, backand a ionic.

Všetky moduly sú do projektu importované pomocou HTML tagov `script` a `link` v hlavičke indexového súboru. Aby boli skripty spustiteľné je nevyhnutné ich deklarovať aj v súbore `app.js`, ktorý zabezpečuje inicializáciu všetkých modulov.

```
angular
  .module('carfriend', [
    'ionic',
    'backand',
    'ngCordova',
    'xeditable',
     ...
]);
```

### 4.3.2 Konfigurácia a používanie BaaS

Aby bolo možné používať služby BaaS je potrebné zaregistrovať poskytovateľa v nastaveniach aplikácie. Back& REST volania sa sprístupnia vložením tajného bezpečnostného tokenu[22] do konfiguračnej metódy aplikácie. Unikátny bezpečnostný token na registráciu užívateľov poprípade na anonymné prihlásenie je zverejnený v administratívnom rozhraní na Back&. Komunikácia v reálnom čase využíva websockety[23], ktoré je potrebné takisto nastaviť. Na konfiguráciu BaaS a ďalších modulov je určený súbor `config.js`.

```
function config(BackandProvider) {
    BackandProvider.setAppName('carfriends');
    BackandProvider.setSignUpToken(token);
    BackandProvider.runSocket(true);
};
```

V AngularJS aplikáciách sa k REST API pristupuje zväčša z tovární (factories) využitím `$http` služby. `$http` umožňuje v Angular službách komunikovať so

[22]Bezpečnostný token je elektronický kľúč, ktorý sa používa na overenie identity užívateľa.

[23]Socket (preklad hrdlo, rúra, potrubie) je mechanizmus pre komunikáciu medzi klientom a vzdialeným serverom. Socket.IO je JavaScriptová knižnica využívajúca websockety na zabezpečenie komunikácie v reálnom čase.

vzdialeným serverom vygenerovaním HTTP požiadavky, ktorá vracia *promise*. Vytvorenie a poslanie nového upozornenia na Back& REST rozhranie je načrtnuté v nasledujúcom kóde.

```
function notificationsService(Backand, $http, $q) {
  var baseUrl = '/1/objects/';
  var object = 'notifications/';
  function sendNotification(notif) {
       var deffered = $q.defer();
        $http({
            method: 'POST',
            url: Backand.getApiUrl() + baseUrl + object,
            params: {
                parameters: {
                    email: notif.userEmail
                }
            },
            data: {
              situation: notif.situation,
              ...
            }
        }).then(function (response) {
            deffered.resolve(response);
        }, function (error) {
            deffered.reject(error);
        });
        return deffered.promise;
    };
};
```

Zatiaľ čo `data` sa ukladajú priamo do databázy, `params` obsahujú dodatočné informácie. Emailová adresa posielaná v objekte `parameters` je potrebná pre Back& sockety. Tie sa musia nastaviť na svojich obidvoch stranách. Back& server umožňuje vytvárať vlastný kód, ktorý sa spustí určitou CRUD udalosťou. Po vytvorení nového upozornenia v databáze, server vytiahne z parametrov emailovú adresu cieľového užívateľa a využitím websocketu ho okamžite upozorní. Túto akciu je možné na strane servera reprezentovať nasledujúcim kódom.

```
function backandCallback(userInput, dbRow, parameters, userProfile) {
  if(parameters.email)
   socket.emitUsers("new_notification",userInput, [parameters.email]);
};
```

Na strane klienta stačí načúvať na udalosť `new_notification` a ak je užívateľ prihlásený, tak bude okamžite informovaný o zmene bez nutnosti obnovovania stránky.

```
Backand.on('new_notification', function (data) {
        $rootScope.$broadcast("newNotification", data);
    });
```

### 4.3.3 Stavy aplikácie

Na začiatku je nutné aplikáciu deklarovať na jeden HTML tag v indexovom súbore. V Ionic aplikácii sa všetky šablóny jednotlivých stránok vykresľujú do HTML elementu `body`. Preto je potrebné tomuto elementu definovať atribút `ng-app="carfriend"`.

V aplikácii sa strieda 12 rôznych šablón, pričom každá má vlastný controller (napr. `accountCtrl`, `notificatorCtrl`, `signUpCtrl`, `loginCtrl`) a niektoré controllery sú obecne používané vo viacerých šablónach (napr. `searchCtrl`, `notificationsCtrl`). Controller sa viaže k danej šablóne špecifikáciou atribútu `ng-controller` v rodičovskom HTML elemente. Lepší variant je deklarácia controlleru pri nastavovaní smerovania (routing) medzi jednotlivými stavmi aplikácie. Nastavenie všetkých stavov je uvedené v súbore `routes.js`.

```
$stateProvider
   .state('tab.chat-detail', {
    url: '/chats/:chatId',
    params: {
        user: null
    },
    views: {
        'tab-chats': {
            templateUrl: 'templates/chat-detail.html',
            controller: 'ChatDetailCtrl',
            controllerAs: 'vm'
        }
    }
});
```

Pri špecifikácii stavu sa vždy povinne uvádza jeho názov, url adresa a cesta k šablóne. Ostatné vlastnosti ako parametre alebo controller sú doplňujúce. Uvedený stav je určený pre práve otvorený chat s konverzáciou, ktorej ID je parametrom url adresy. Do tohto stavu sa užívateľ dostane výberom jednej konkrétnej konverzácie v záložke chaty, ktorá obsahuje zoznam všetkých zahájených konverzácií. Pri prechode sa zároveň prenášajú predom získané dáta o užívateľovi, ktorý je súčasťou tejto konverzácie. Iniciátorom prechodu môže byť controller alebo odkaz v šablóne.

```
function goToChat(chatId, data){
  return $state.go('tab.chat-detail', {chatId: chatId, user: data});
};
```

### 4.3.4 Základne funkcie

#### Autentizácia

Autentizáciu užívateľov zabezpečuje samotný Back&, ktorý má predpripravené funkcie na prihlásenie, odhlásenie, registráciu užívateľov a obnovu hesla. Pri registrácii Back& kontroluje unikátnosť emailovej adresy, pri prihlasovaní zas správnosť hesla. V aplikácii je vytvorená továreň `AuthService`, ktorá vracia všetky autentizačné funkcie a zhrňuje údaje o prihlásenom užívateľovi. Prihlásenie užívateľa má na starosti `LoginCtrl`, ktorý kontroluje úplnosť dát a následne volá funkciu `signIn` z autentizačnej továrne. Jej obsah je možné skrátiť na jeden riadok.

```
function signIn(username, password) {
  return Backand.signin(username, password);
};
```

Prihlásený užívateľ sa uloží do premennej `currentUser` a jeho vlastnosti nie je nutné dokola načítavať, ale stačí vždy zavolať `AuthService.getUser()`.

#### Lokalizačné služby

Lokalizačné služby sa v aplikácii využívajú na viacerých miestach, a preto je vhodné vytvoriť továreň, ktorá tieto služby zoskupuje na jedno miesto. Továreň `geolocationService` vracia tri funkcie:

1. `getCurrentPosition()` – zistenie GPS súradníc aktuálnej polohy,
2. `getCurrentLocation(position)` – preklad polohy do slovného vyjadrenia,
3. `isStreetExist(street)` – kontrola existencie ručne vloženej ulice a vrátenie všetkých výsledkov.

Prvá funkcia využíva modul ngCordova, ostatné funkcie využívajú API, ktoré poskytujú Google mapy.

#### Zabezpečenie vozidla

Táto funkcia je určená pre majiteľov vozidiel a je im sprístupnená v záložke garáž. Vodič po jej spustení získava kontrolu o dianí na danej ulici a v jej okolí. Inými slovami, aplikácia bude prijímať všetky upozornenia v oblasti parkovania. Túto funkciu ovláda `AccountCtrl`. Na jej správnu funkčnosť je potrebné poznať lokalizáciu vozidla. Tá môže byť zistená dvoma spôsobmi:

1. `insertStreetByGPS()` – lokalizácia bude zistená využitím lokalizačného systému zariadenia alebo
2. `insertStreetManually(street)` – zariadenie užívateľa blokuje alebo nedisponuje lokalizačným systémom, a preto je potrebné zadať názov ulice ručne.

Prvá funkcia využíva lokalizačnú funkciu `getCurrentPosition()` a v druhej je potrebné overiť existenciu zadanej ulice zavolaním lokalizačnej funkcie

`isStreetExist(street)`. V pozitívnych prípadoch sú zistené GPS súradnice polohy a slovný názov ulice. Obidve informácie sú potrebné a sú uložené do objektu `vm.parkingPosition`. Následne je zavolaná funkcia `secureMe()`, ktorá vykoná tri činnosti. Prvá činnosť uloží do databázy GPS polohu vozidla, druhá skontroluje, či sa na danej ulici nekoná blokové čistenie a tretia vypíše históriu nehôd, pokút a priestupkov na danej ulici.

```
function secureMe() {
        var promises = [];
        promises.push(saveCurrentLocation());
        promises.push(checkCleaning());
        promises.push(checkStreetHistory());
        $q.all(promises).then(function () {
            ...
        });
};
```

Do dátového typu pole sa pridajú tri asynchrónne operácie, ktorých úspešné vykonanie spustí časový odpočet symbolizujúci dobu zabezpečenia vozidla. Táto doba je pre bežného užívateľa stanovená na pol hodinu a pre prémiového užívateľa je jej dĺžka neobmedzená. Po jej skončení sa užívateľovi preruší ochrana a nebude môcť prijímať informácie o dianí v okolí vozidla.

### Poslanie upozornenia

Kontrolu nad posielaním upozornení má `NotificatorCtrl`, ktorý okrem toho, že kontroluje úplnosť vstupných dát, tak zároveň obaľuje upozornenie doplňujúcimi informáciami o emailovej adrese cieľového vodiča, emailových adresách blízkych vodičov a pristupuje k natívnym funkciám pri zaznamenávaní GPS súradníc a tvorbe fotografií. Na vytvorenie fotografie slúži funkcia `makePhoto()`, ktorá využíva na prístup k fotoaparátu metódu `getPicture(options)` z modulu ngCordova.

```
function makePhoto() {
    var options = {
        quality: 80,
        sourceType: Camera.PictureSourceType.CAMERA,
        saveToPhotoAlbum: false,
        ...
    };
    $cordovaCamera.getPicture(options).then(function (iData) {
        vm.notification.photo = "" + iData;
    }, function (err) {
        vm.photoError = err;
    });
};
```

Zaparkované vozidlá v okruhu od určitej polohy je možné nájsť dotazom na databázu, ktorý má dva vstupné parametre. Prvým je dvojica GPS súradníc, latitude a longitude, a druhým je rozsah v metroch. Parametrom `deep` sa vrátia nie len vozidlá, ale aj ich majitelia, ktorých email je potrebným vstupným parametrom pre websockety.

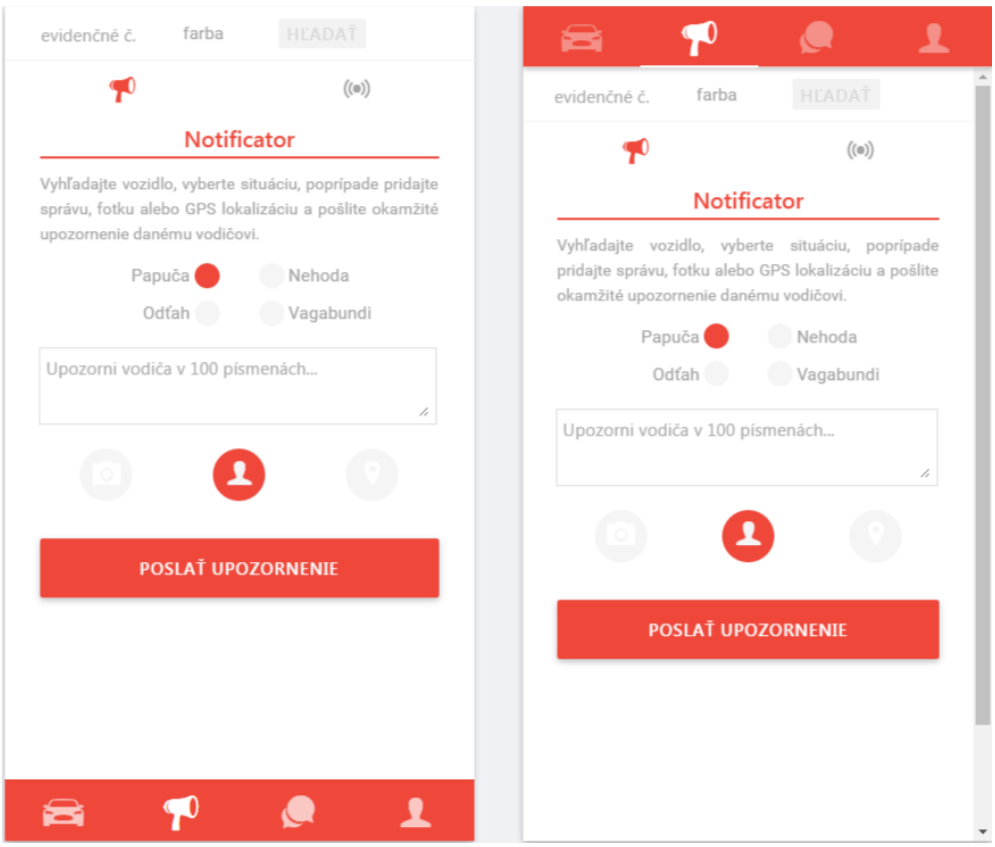
```
$http({
    method: 'GET',
    url: Backand.getApiUrl() + baseUrl + 'parking/',
    params: {
      deep: true,
      filter: {
       "q": {
          "geo": {"$within": [[pos.latitude, pos.longitude], within]}
           }
      }
    }
})
```

### Užívateľské rozhranie

Užívateľské rozhranie webovej aplikácie je vytvárané kombináciou technológií HTML a CSS. Mobilný rozmer definuje ďalšiu vlastnosť aplikácie – responzívnosť. Responzívny rámec vytvára framework Ionic, ktorý je dopĺňaný vlastnými responzívnymi elementami. Tie boli vytvorené najčastejšie pomocou CSS kontajneru flexbox. Flexbox je výnimočný svojou pružnosťou, vďaka ktorej automaticky vypĺňa priestor, a tým odbremeňuje vývojára od prácneho počítania polohy a umiestnenia prvkov. Nasledujúce SCSS pravidlo zobrazí obsah elementu s triedou userInfo v riadku, kde jednotlivé prvky sú zarovnané vedľa seba a čo najviac do stredu.

```
userInfo {
    @include display-flex();
    @include flex-direction(row);
    @include justify-content(center);
}
```

Kostra užívateľského rozhrania vychádza zo záložkovej šablóny, ktorá bola zvolená pri vytváraní Ionic projektu. Z prvej záložky sa spúšťajú akcie týkajúce sa vozidla, v druhej záložke sa vytvárajú a posielajú upozornenia, v tretej záložke je implementovaný chat a štvrtá záložka zoskupuje informácie o užívateľovi, o jeho aktivitách a obsahuje nastavenia aplikácie. Na obrázku 10 sú odfotené úvodné obrazovky na Android a iOS zariadeniach. Ďalšie ukážky sú vložené do príloh tejto práce.




Obrázok 10: Ukážka užívateľského prostredia na iOS a Android platforme

Možnosti používania aplikácie sa líšia užívateľským typom. Preto je potrebné meniť užívateľské rozhranie na základe typu užívateľa. Je zrejmé, že užívateľ bez vozidla nebude môcť zabezpečiť vozidlo alebo prijímať upozornenia. Tieto možnosti mu teda musia byť buď v rozhraní skryté alebo takýto užívateľ musí mať vlastné užívateľské rozhranie. Pretože AngularJS ponúka prepracované direktívy na skrývanie resp. zobrazovanie HTML elementov bola zvolená prvá možnosť. Na základe toho, či má užívateľ vozidlo alebo nie, je možné jednoducho pomocou direktívy `ng-show` zobraziť užívateľovi garáž s jeho vozidlom alebo naopak formulár na pridanie vozidla.

### 4.3.5 Testovanie aplikácie

Hlavná časť testovania prebiehala počas implementácie, ktorá odhalila prirodzene najviac chýb. Následne prebehli testy aj v teréne. Počas implementácie sa testovalo vo vývojárskych nástrojoch prehliadača Chrome a v mobilnom zariadení Android. Na prenos aplikácie do zariadenia sa využívala aplikácia Ionic View, ktorá bola už bližšie opísaná. Testovanie počas implementácie spočívalo v hľadaní chýb na základe vlastných podnetov a pokusov. Na skúšobných dátach boli demonštrované všetky funkcie aplikácie. Tento test nie je dostatočný na to, aby odhalil chyby v reálnych situáciách.

Aplikácia bola vyskúšaná v reálnych zariadeniach, konkrétne Android 6.0, Android 5.1 a iOS 9.3. Na týchto zariadeniach sa vykonala registrácia užívateľa poprípade vozidla a následne prebehlo prihlásenie. Tieto operácie prebehli na skúšaných systémoch bez problémov. Medzi týmito zariadeniami sa vykonala simulácia vzájomnej výmeny upozornení a správ. Upozornenia zväčša dorazili okamžite a v poriadku. Občasným problémom bolo, že aplikácia neupozornila užívateľa o novom upozornení. To vyskočilo až v momente, keď užívateľ otvoril záložku, v ktorej sa tieto upozornenia zobrazujú. Pravdepodobne nastal problém v komunikácii v reálnom čase, kedy buď aplikácia nezachytila volanie backendovej služby alebo služba toto volanie ani nevyslala. Testom spoľahlivo prešli natívne funkcie, ktoré bez problémov zachytili fotografiu alebo lokalizovali užívateľa. Kontrola blokového čistenia a história priestupkov na danej ulici fungovala takisto správne. Pri blokových čisteniach je potrebné pokračovať v zbieraní informácií o termínoch v ďalší častiach mesta Brno aby mohla aplikácia pokrývať čo najväčšie územie.

Vzhľad a užívateľské rozhranie fungovalo takmer správne na každom testovanom zariadení. Na obrázku 11 je možné sledovať chybu v zobrazovaní na zariadení iPhone. Na týchto zariadeniach sa síce správne presúva navigácia na spodok obrazovky, ale tým vytláča ostatne prvky smerom hore do stavového riadka telefónu.


Obrázok 11: Chyba zobrazenia v zariadení iPhone

Testovanie dokázalo, že aplikácia je funkčná avšak nie je ešte plne pripravená pre užívateľov. Je zrejmé, že potrebuje rozsiahlejšie testovanie, lepšie odladenie a optimalizáciu pre staršie verzie operačných systémov.

# 5 Diskusia

V natívnych obchodoch a na internete je množstvo užitočných aplikácií, ktoré sa snažia užívateľom niečo uľahčiť. Ich počty prekračujú možnosti ľudského vnímania, častokrát nie sú jedinečné a nezískajú si ani obľubu medzi užívateľmi. Z toho dôvodu, ak sa nevyvíja firemná aplikácia, je veľmi dôležitý nápad a princíp aplikácie. V tomto duchu prebiehal vývoj aj tejto aplikácie. Jej cieľom je vyplniť medzeru na trhu s dopravnými aplikáciami, ktorých množstvo takisto narastá. Vo všeobecnosti sa dopravné aplikácie zameriavajú na navigáciu vozidla, hlásenie policajných radarov a kontrol, zaznamenávanie servisných úkonov a nákladov, vyhľadávanie lacných čerpacích staníc, vypočítanie výšky pokuty, odvozy a taxi služby, hľadanie voľných parkovacích miest a iné.

Využívaním evidenčných čísel vozidla sa otvára nový spôsob komunikácie medzi ľuďmi. Evidenčné číslo musí mať každé vozidlo, je jedinečné, verejné a táto aplikácia dokazuje, že je ho možné využiť aj iným, ako predurčeným, spôsobom. Okrem toho, že umožňuje presne adresovať upozornenie, poskytuje možnosť identifikácie užívateľa, nadviazania komunikácie a možnosť sociálneho zblíženia. Čo keby bolo evidenčné číslo vozidla ďalším kontaktným údajom, podobne ako telefónne číslo?

Pri implementácii boli použité odporúčané moderné technológie a programátorské postupy, ktoré boli naštudované zo súčasnej literatúry a spoľahlivých internetových návodov a diskusií. Aktuálnosť a stabilita bola pri vývoji prvoradá. Do aplikácie sa nepodarilo pridať všetky funkcie a splniť vytýčené požiadavky. Tie, ale nezanikajú a budú spolu s ďalšími predmetom pokračujúcej implementácie. Základné funkcie začínajúce od samotnej registrácie cez posielanie upozornení v reálnom čase až po prácu s lokalizačnými službami boli úspešne zahrnuté do aplikácie. Aplikáciu je možné používať.

Aplikácia bude dostupná pre verejnosť až po opravení nedostatkov implementácie a zahrnutí nedokončených funkcií. Okrem natívnych obchodov je plánované ju umiestniť aj online.

## 5.1 Nedostatky implementácie

Najväčším nedostatkom hotovej implementácie je absencia platobného systému, ktorý bolo plánované zahrnúť do aplikácie. Nakoľko P2P platby sú stále vo vývoji a nie sú ešte bežne dostupnou záležitosťou, tak ich zavedenie je spojené s množstvom obmedzení. Prvé a najväčšie obmedzenie je nedostupnosť potrebných služieb v Európe alebo len v Českej a Slovenskej republike. Druhým obmedzením niektorých dostupných systémov je nutnosť registrácie a používania ich vlastnej aplikácie. Tretie obmedzenie je zložitejší variant systému, pri ktorom dochádza k rozdeleniu jednej čiastky medzi aplikáciou a užívateľom. Po komunikácii so zamestnancom spoločnosti Braintree bolo zistené, že vyhovujúca a vhodná služba Marketplace nie je a v najbližšej dobe ani nebude v Európe k dispozícii.

Tento zamestnanec ale poradil, ako je možné túto situáciu vyriešiť. Riešenie spočíva v uskutočnení jednej platby v dvoch krokoch. Celá platba by najskôr smerovala z účtu vodiča na účet aplikácie, kde by prebehlo jej rozdelenie a určitá čiastka by následne bola manuálne preposlaná na účet cieľového užívateľa. Tým by však platba neprebehla okamžite, ale odstupom pár dní. Druhým riešením, ktoré zamestnanec poradil, je nakontaktovať PayPal ohľadom ich produktu Adaptive Payments. To je v najbližšej dobe v pláne.

Tým, že sa nepodarilo implementovať platby, tak nie je možné ani rozlišovať užívateľov na bežných a prémiových. Preto má zatiaľ každý užívateľ aplikácie rovnaké právomoci. Na push notifikácie, ktoré upozorňujú užívateľov priamo na obrazovke sa bude musieť ešte chvíľu počkať. Zatiaľ čo Back& aktívne pracuje na svojej vlastnej implementácii tejto služby, tak zároveň ponúka ako alternatívu integráciu push služby PushWoosh alebo Netmera. Tieto služby boli vyskúšané ale nepracovali tak ako mali, a preto boli z aplikácie odstránené.

Nedostatkom aplikácie býva občasný slabý výkon alebo chyby v histórii navigovania, kedy nie je možné sa stláčaním spätného tlačidla na Android zariadeniach dostať na začiatok poprípade aplikáciu úplne zavrieť. Problém predstavujú aj nezachytené chyby, pri ktorých aplikácia prestane reagovať a užívateľ nevie, čo sa deje. Tieto chyby je potrebné nájsť a následne vyladiť.

## 5.2 Možné vylepšenia

Aplikáciu je možné samozrejme ďalej rozširovať. Napríklad kontrolu čistenia ulice by bolo vhodné spojiť s navigáciou na najbližšiu ulicu, na ktorej sa čistenie už nekoná. Nevyhnutné je pridanie možnosti predaja vozidla a s tým spojené prepísanie práv inému užívateľovi. Pri skutočnom nasadení aplikácie je potrebné zaviesť ochranu pred registráciou vozidla, ktoré užívateľ nevlastní.

Užívateľov, ktorí schválne posielajú nepravdivé a spamerské upozornenia by bolo vhodné v aplikácii nahlasovať. Zaujímavou funkciou by bola aj navigácia vodiča k zabudnutému zaparkovanému vozidlu. Z údajov o priestupkoch a pokutách by bolo možné štatisticky deliť ulice na nebezpečné a pokojné. Namiesto lístkov za stieračmi by mohli majitelia zablokovaného pozemku, samotná odťahovacia služba poprípade policajti poslať informáciu vodičovi priamo cez aplikáciu. Niektorí užívatelia by možno tiež uvítali zaznamenávanie histórie navštívených lokalít alebo možnosť registrácie viacerých vodičov na jedno vozidlo.

Pre obchodníkov s autami by bola zrejme užitočná funkcia, na hľadanie predávaných áut. Tá by mohla fungovať spôsobom, že majiteľ by v aplikácii označil vozidlo ako predávané a záujemcom v okolí by sa zobrazila jeho ponuka.

# 6 Záver

Cieľom práce bolo navrhnúť a následne implementovať aplikáciu, ktorá spája účastníkov cestnej premávky prostredníctvom rýchlych upozornení, zachraňuje majiteľom peňaženky poprípade vozidlá, poskytuje priestor na sociálnu interakciu a tiež ponúka možnosť peňažného zárobku. Pred voľbou implementačných nástrojov sa hľadal spôsob ako v čo najväčšom rozsahu pokryť množstvo potenciálnych užívateľov. Práve preto sa vývoj vybral hybridnou cestou, ktorá priniesla určité výhody, ale taktiež nastražila mnohé prekážky.

Práca priniesla porovnanie prístupov k tvorbe aplikácií spustiteľných na mobilných zariadeniach a hlbšie preštudovala zvolený hybridný vývoj. Práca zistila, že je o tento spôsob vývoja záujem a podľa všetkého ešte v budúcnosti porastie. Práca sa tiež zamerala na nástroje, ktoré vytvárajú natívne užívateľské rozhranie a porovnala rozdiely zobrazovania na Android a iOS platformách. Zistené výhody služby ako serverovej možnosti boli zužitkované v implementovanej aplikácii a tiež sú vhodným zdrojom informácií pre nerozhodného vývojára. Počas vývoja sa totižto zvolená serverová alternatíva javila ako plnohodnotná náhrada tradičného serverového prístupu. Práca čiastočne načala tému NoSQL databáz, ktoré sú ešte skôr novinkou ako bežne zaužívaným databázovým systémom. V rámci tejto bakalárskej práce boli načrtnuté taktiež aj P2P platby, ktoré predstavujú budúcnosť platobného styku. Je teda možné konštatovať, že táto práca prináša čitateľovi zužitkovateľné poznatky.

Praktická časť práce sa zaoberala samotným návrhom a implementáciou aplikácie. Navrhnutý dátový a funkčný model boli základom pre nasledujúci vývoj. Ten priblížil prácu vo frameworku AngularJS, Ionic a Apache Cordova. Práca poskytuje návod na zahájenie vývoja v Ionic frameworku vrátane popisu základných ovládacích príkazov. V práci bolo predvedené, ako ľahko je možné napojiť aplikáciu na službu a ako je možné s touto službou komunikovať. Ďalej práca vysvetľuje postup a spôsob tvorby databázy pomocou JSON modelu. Tieto princípy je možné s malou zmenou aplikovať na všetkých poskytovateľov serverových služieb. V diskusnej časti boli opomenuté nedokončené a chybné časti implementácie. Práca v tejto časti prišla s možnými vylepšeniami a ďalšími funkciami vhodnými pre túto aplikáciu.

Aby sa prejavili prínosy tejto aplikácie je ju potrebné v hotovej podobe rozšíriť medzi vodičov. Čím je väčšia užívateľská základňa, tým je tento druh aplikácie prínosnejší. Veľkou výhodou aplikácie je bezplatná registrácia, ktorou nemôže majiteľ vozidla nič stratiť, ale len získať. Medzi prínosy aplikácie je možné zahrnúť aj s odvolaním sa na výsledky dotazníkového šetrenia zvýšenie bezpečnosti vozidla. Tá sa dosiahne väčším prehľadom o vozidle. Pre vodičov a majiteľov vozidiel je prínosom aplikácie možnosť ušetriť peniaze, vyhnúť sa pokute, priestupku alebo škode. Aj na základe dotazníkového šetrenia je možné tvrdiť, že veľkým lákadlom pre užívateľov je možnosť peňažnej odmeny za svoju aktivitu. I keď sa túto požiadavku nepodarilo do tohto momentu naplniť, i tak ostáva prioritou v pokračovanej implementácii. Nie len pre užívateľov ale aj pre policajtov poprípade mestské samosprávy je prínosom

aplikácie databáza s priestupkami, na základe ktorej je možné následne štatisticky vyhodnocovať bezpečnosť a frekvenciu priestupkov v rámci jednotlivých ulíc.

Táto práca splnila vytýčené ciele a je ju možné vyskúšať na Android a iOS zariadeniach alebo prostredníctvom webového prehliadača. Postup spolu so zdrojovými kódmi a spustiteľnými súbormi je možné nájsť na priloženom CD.

# 7 Referencie

ADAPTIC. *Backend* [online]. [cit. 2016-4-21]. Dostupné na: http://www.adaptic.cz/znalosti/slovnicek/backend/.

APPEL, R. *Modern Apps : Mobile Web Sites vs. Native Apps vs. Hybrid Apps* [online]. 2014 [cit. 2016-3-20]. Dostupné na: https://msdn.microsoft.com/en-us/magazine/dn818502.aspx.

BOČEK, J., ŠULEK, M., CIBULKA, J. *Mapa odtahů v Brně: Objevte nejrizikovější místa* [online]. 2014-11-25 [cit. 2016-5-10]. Dostupné na: http://www.rozhlas.cz/zpravy/data/_zprava/1424350.

BRADY, J. *How Forge works and why we're proud of it* In: Blogger [online]. 2012-1-26 [cit. 2016-3-24]. Dostupné na: http://trigger.io/cross-platform-application-development-blog/2012/01/26/how-forge-works-and-why-were-proud-of-it.

DRIFTY CO. *What is Ionic?* [online]. 2016 [cit. 2016-17-3]. Dostupné na: https://github.com/driftyco/ionic.

DUNN, C. *Why Trigger.io doesn't use PhoneGap – 5x faster native bridge* In: Blogger [online]. 2012-2-24 [cit. 2016-3-24]. Dostupné na: http://trigger.io/cross-platform-application-development-blog/2012/02/24/why-trigger-io-doesnt-use-phonegap-5x-faster-native-bridge/.

FINK, G., FLATOW, I. *Pro single page application development: using Backbone.js and ASP.NET.* New York: Apress, 2014. 324 s. ISBN 978-1-4302-6673-0.

FOWLER, A. *Nosql for dummies.* New Jersey: John Wiley & Sons, 2015. 456 s. ISBN 978-1-118-90578-4.

GOOGLE TRENDS. *Záujem v priebehu času* [online]. [cit. 2016-5-5]. Dostupné na: https://www.google.sk/trends/explore#q=%22hybrid+mobile+application%22.

GOOGLE INC. *Android System WebView* [online]. 2016-5-2 [cit. 2016-5-5]. Dostupné na: https://play.google.com/store/apps/details?id=com.google.android.webview&hl=sk.

HORÁKOVÁ, V. *SMS zachrání řidiče před odtahem, upozorní na blokové čištění ulic* [online]. 2015-2-5 [cit. 2016-4-21]. Dostupné na: http://brno.idnes.cz/ridice-v-cesku-zacinaji-upozornovat-na-blokove-cisteni-sms-zpravy-10w-/brno-zpravy.aspx?c=A150204_2136839_brno-zpravy_daj.

ICT REVUE. *Cloudové služby se rozmnožily aneb co všechno je dnes v cloudu* [online]. 2015-4-22 [cit. 2016-4-21]. Dostupné na: http://ictrevue.ihned.cz/c1-

63904590-cloudove-sluzby-se-rozmnozily-aneb-co-vsechno-je-dnes-v-cloudu.

In: Wikipedia: the free encyclopedia. *Web application* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, last modified on 14. 12. 2015 [cit. 2015-12-14]. Dostupné na: https://en.wikipedia.org/wiki/Web_application.

In: Wikipedia: the free encyclopedia. *Škálovatelnost* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, last modified on 25. 7. 2013 [cit. 2016-5-5]. Dostupné na: https://cs.wikipedia.org/wiki/%C5%A0k%C3%A1lovatelnost.

Kovalyov, A. *Beautiful Javascript Leading Programmers Explain How They Think.* California: O'Reilly Media, 2015. 168 s. ISBN 978-1-449-37075-6.

Kovács, J., Sivák, L. *Vývoj mobilných aplikácií v PhoneGap-e* [online]. 2013 [cit. 2016-20-3]. Dostupné na: http://www.buckleup.sk/sk/nove/vyvoj-mobilnych-aplikacii-phonegap.

McPeak, J. *Beginning JavaScript.* 5. vyd. Indianapolis: Wiley, 2015. 324 s. ISBN 978-1-118-90333-9.

Mikowski, M., Powell, J. *Single page web applications: JavaScript end-to-end.* New York: Manning Publications, 2014. 432 s. ISBN 978-1-6172-9075-6.

Mitchel, J. *PHP web services: [APIs for the modern web].* California: O'Reilly Media, 2013. 118 s. ISBN 978-1-449-35656-9.

Monteiro, F. *Learning Single Page Web Application Development.* Birmingham: Packt Publishing, 2014. 207 s. ISBN 978-1-78355-209-2.

Morony, J. *8 Reasons Why I'm Glad I Switched to the Ionic Framework* [online]. 2015-9-18 [cit. 2016-3-17]. Dostupné na: http://www.joshmorony.com/8-reasons-why-im-glad-i-switched-to-the-ionic-framework/.

Morony, J. *A Summary of Backend Options for HTML5 Mobile Applications* [online]. 2016-2-2 [cit. 2016-21-4]. Dostupné na: http://www.joshmorony.com/a-summary-of-backend-options-for-html5-mobile-applications/.

Mozilla Developer Network. *JavaScript* [online]. 2015-12-6 [cit. 2015-12-10]. Dostupné na: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

Mrozek, J. *Moderní vývoj aplikací* In: Blogger [online]. 2013-5-2 [cit. 2015-12-9]. Dostupné na: http://weblog.ronnieweb.net/2013/05/moderni-vyvoj-aplikaci/.

Mrozek, J. *Začínáme s AngularJS* [online]. 2012-11-30 [cit. 2016-5-9]. Dostupné na: https://www.zdrojak.cz/clanky/zaciname-s-angularjs/.

O'Sullivan, Ch. *A Tale of Two Platforms: Designing for Both Android and iOS* [online]. 2015-4-15 [cit. 2016-3-18]. Dostupné na: http://webdesign.tutsplus.com/articles/a-tale-of-two-platforms-designing-for-both-android-and-ios--cms-23616.

Panda, S. *An Overview of JavaScript Promises* [online]. 2013-12-25 [cit. 2016-5-14]. Dostupné na: http://www.sitepoint.com/overview-javascript-promises/.

Panhale, M. *Beginning Hybrid Mobile Application Development.* New York: Apress, 2016. 244 s. ISBN 978-1-484213-15-5.

Papa, J. *Angular 1 Style Guide* [online]. 2016-3-31 [cit. 2016-4-21]. Dostupné na: https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md.

Pehlivanian, A., Nguyen, D. *JavaScript Okamžitě.* Preložil Ondřej Baše. Brno: Computer Press, 2014. 160 s. ISBN 978-80-251-4163-2.

PhoneGap. *Adobe PhoneGap Build* [online]. [cit. 2016-3-22]. Dostupné na: http://docs.build.phonegap.com/.

PhoneGap Blog. *PhoneGap, Cordova, and what's in a name?* [online]. 2012-3-19 [cit. 2016-3-22]. Dostupné na: http://phonegap.com/2012/03/19/phonegap-cordova-and-what\%E2\%80\%99s-in-a-name/.

Powell, J. *What Titanium Appcelerator REALLY is and How it Works!* [online]. 2012 [cit. 2016-21-3]. Dostupné na: http://forumone.com/blog/what-titanium-appcelerator-really-and-how-it-works/.

Remick, J. *What Is a Web App? Here's Our Definition* [online]. 2011-1-30 [cit. 2015-12-14]. Dostupné na: http://nordicapis.com/why-you-should-build-apps-with-an-api-backend-baas/.

Riggins, J. *Why You Should Build Apps With An API Backend – BaaS* [online]. 2015-3-16 [cit. 2016-4-21]. Dostupné na: http://nordicapis.com/why-you-should-build-apps-with-an-api-backend-baas/.

Stieben, D. *5 Reasons Why You Should Make Your Own Server* [online]. 2012-1-21 [cit. 2016-4-21]. Dostupné na: http://www.makeuseof.com/tag/5-reasons-server/.

Suehring, S. *JavaScript: Krok za krokem.* Preložil Jakub Zemánonek. Brno: Computer Press, 2008. 335 s. ISBN 978-80-251-2241-9.

TCL DigiTrade. *Notes data na Android mobilu … i offline* [online]. 2016-1-4 [cit. 2065-4-21]. Dostupné na: http://www.tcl-digitrade.cz/2016/01/04/notes-data-na-android-mobilu-i-offline/.

TRIGGER.IO. *Trigger.io Docs* In: Quickstart [online]. [cit. 2016-3-22]. Dostupné na: https://trigger.io/docs/current/index.html.

TRIGGER.IO. *Trigger.io Pricing Plans* [online]. [cit. 2016-3-24]. Dostupné na: https://trigger.io/pricing/).

VÁCLAVÍK, J. *Vyvíjíme hybridní aplikace v Ionicu: Úvod a instalace* [online]. 2015-7-20 [cit. 2016-4-10]. Dostupné na: https://www.zdrojak.cz/clanky/vyvijime-hybridni-aplikace-v-ionicu/.

WILKEN, J., BRADLEY, A. *Ionic in action: hybrid mobile apps with Ionic and AngularJS.* New York: Manning Publications, 2015. 254 s. ISBN 9978-1-63343-008-2.

XAMARIN. *Xamarin Documentation* [online]. [cit. 2016-21-3]. Dostupné na: https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/.

ZAKAS, NICHOLAS Z. *Professional JavaScript for web developers.* 3. vyd. Indianapolis: Wiley, 2012. 960 s. ISBN 978-1-118-02669-4.

# Prílohy

# A Dotazník a jeho vyhodnotenie

## A.1 Popis

Carfriend je aplikácia, ktorá na základe podnetov od dopravných účastníkov upozorňuje majiteľov vozidiel, že sa niečo nežiadúce deje s ich štvorkolesovými miláčikmi. Upozornenia sa môžu týkať rôznych dopravných situácií (parkovací a jazdný problém, krádež, blokové čistenie atď).

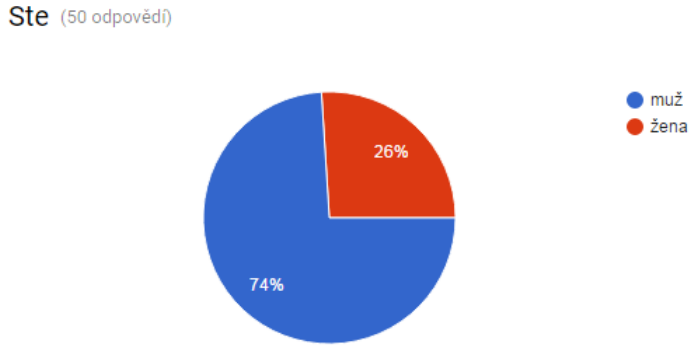
Ak si **vodič**, tak si predstav situáciu, že napríklad zaparkuješ vozidlo na parkovisku pri obchodnom dome, niekto ti ho nabúra a z miesta činu odíde. Tebe stačí byť zaregistrovaný v aplikácii a očitý svedok ti môže poslať upozornenie o búračke spolu s informáciami, napr. o ŠPZ vinníka. Ako poďakovanie za cenné informácie a s tým spojené ušetrenie času, problémov a peňazí budeš môcť poslať nahlasovateľovi odmenu (peniaze, body, slovne poďakovať, a iné). Môžeš byť upozornený o odťahovaní vozidla, papuči na kolese, problémovom parkovaní, pokazených svetlách, krádeži, pohybu nebezpečných osôb, blokovom čistení a množstve ďalších dopravných problémoch.

Ak si **nahlasovateľ**, tak posielaním upozornení môžeš vďaka tejto aplikácii zarobiť a zároveň pomôcť iným. Každé tvoje hlásenie môže byť určitou formou odmenené (peňažne, bodmi, slovne). Poskytnutím relevantných a cenných informácií si buduješ dôveryhodnosť, získavaš nových priateľov a zarábaš. A pri tom zaslanie jedného upozornenia ti zaberie menej než 30 sekúnd.

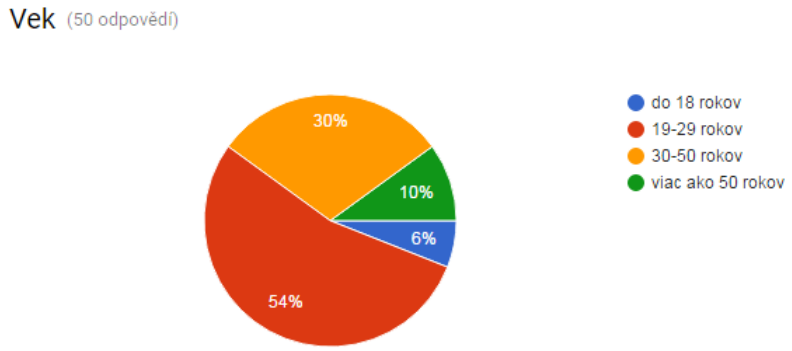
**Prémiové funkcie**

- nebezpečná ulica – upozorňovanie na parkovanie na ulici, kde často dochádza k pokutám a nehodám,
- kontrola parkovania – užívateľ jedným kliknutím zistí, či na aktuálnej ulici neprebieha blokové čistenie alebo iné parkovacie obmedzenie,
- predpoveď pokuty – aplikácia upozorní majiteľa vozidla na premnožené pokuty v oblasti, kde zaparkoval svoje vozidlo,
- možnosť zadať viac vozidiel a často navštevovaných adries,
- kontrola EK a TK.

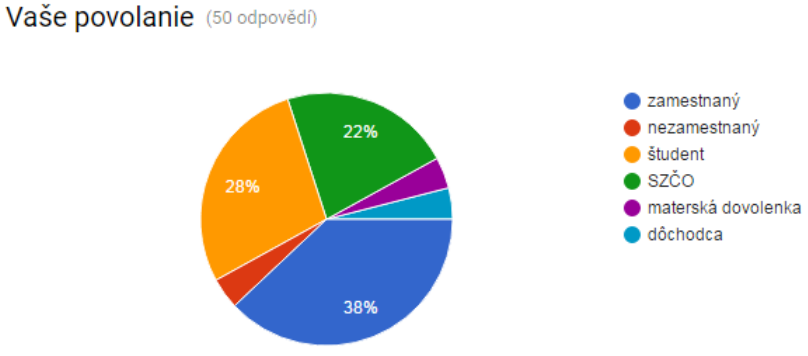
## A.2 Vyhodnotenie



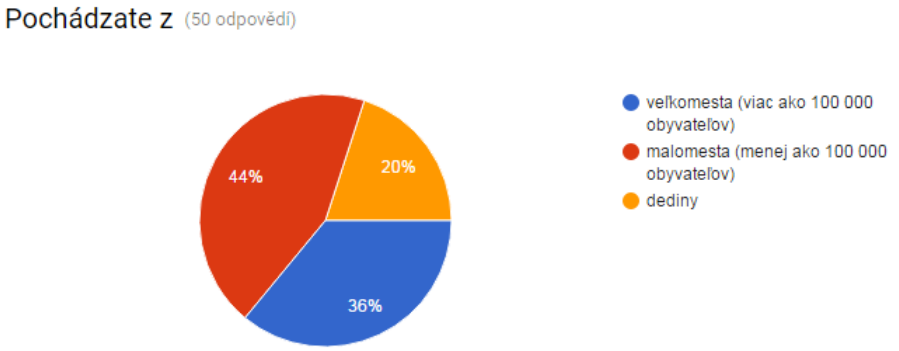

Obrázok 12: Otázka č.1



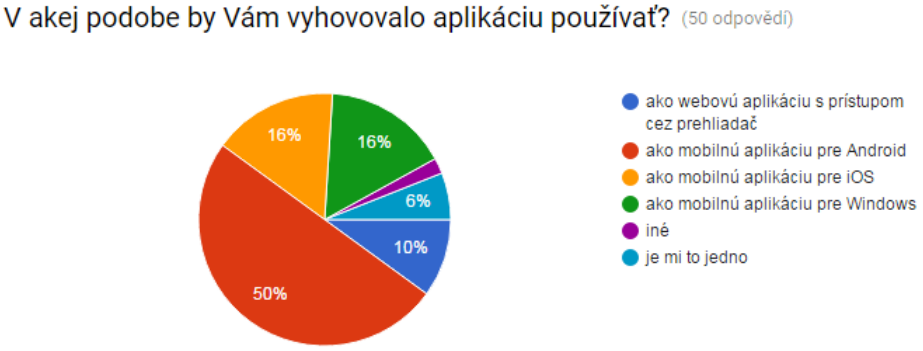

Obrázok 13: Otázka č.2



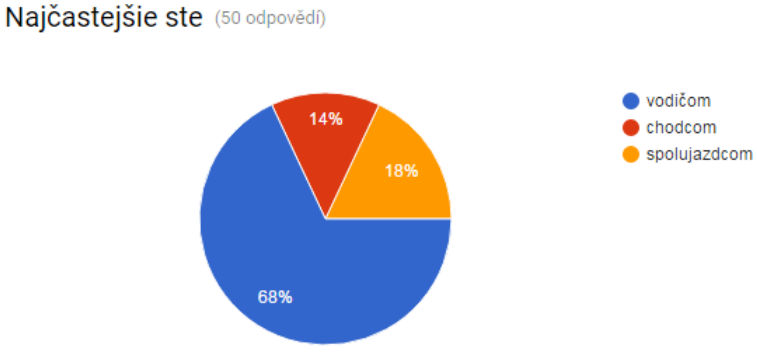

Obrázok 14: Otázka č.3




Obrázok 15: Otázka č.4



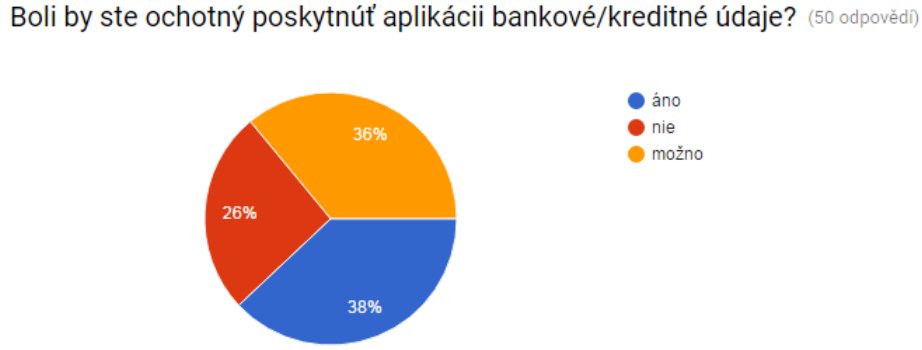

Obrázok 16: Otázka č.5



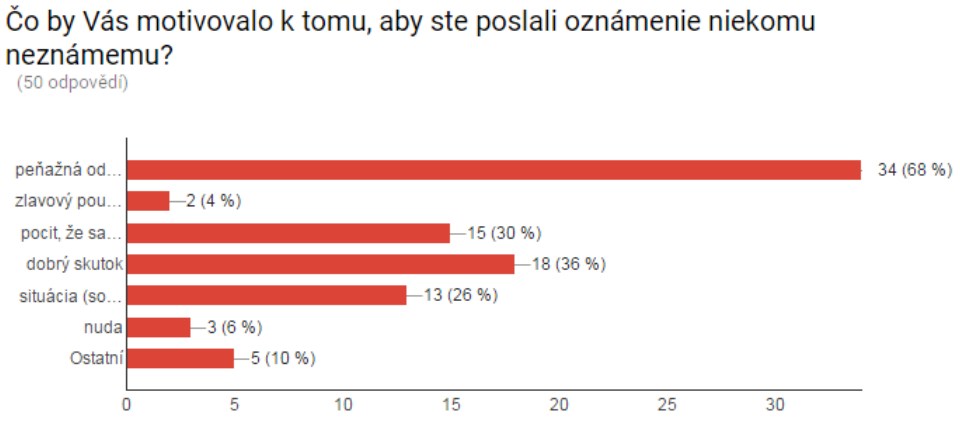

Obrázok 17: Otázka č.6


Obrázok 18: Otázka č.7


Obrázok 19: Otázka č.8

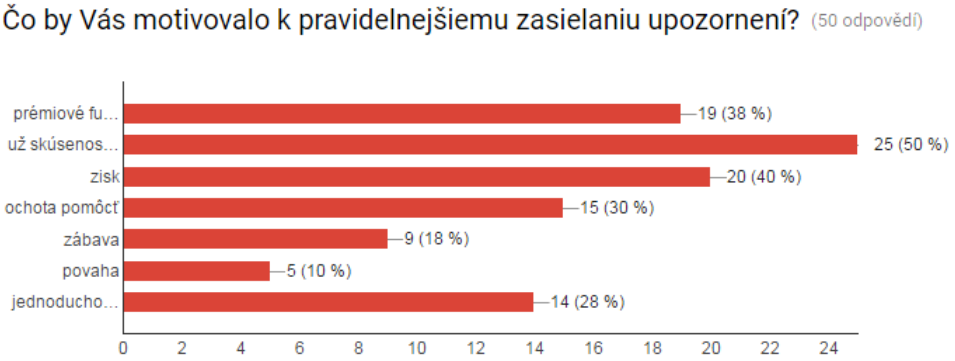



Obrázok 20: Otázka č.9

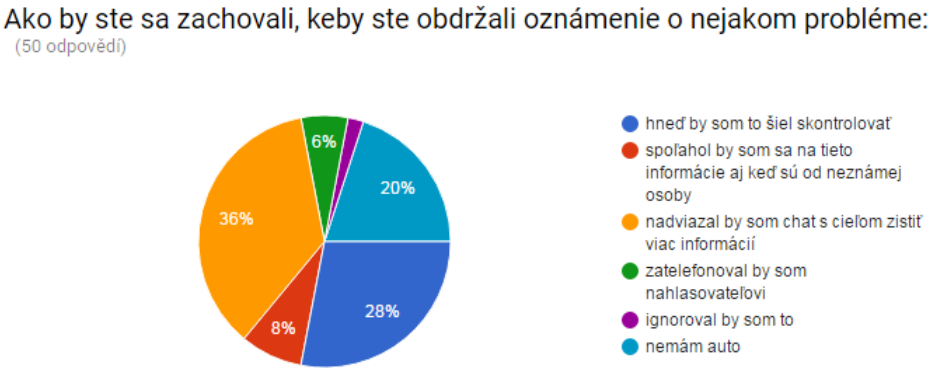



Obrázok 21: Otázka č.10

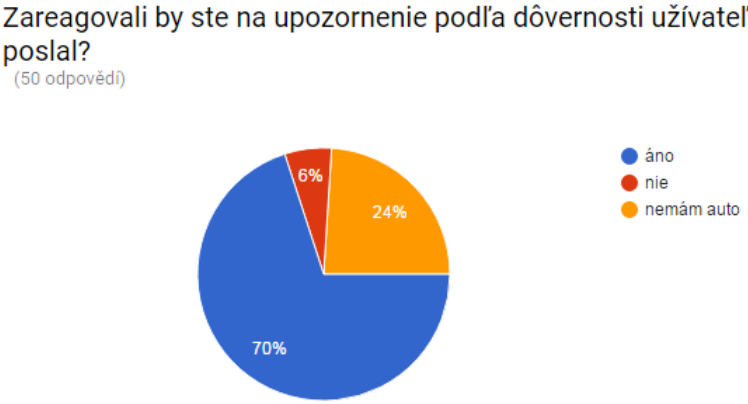

Obrázok 22: Otázka č.11



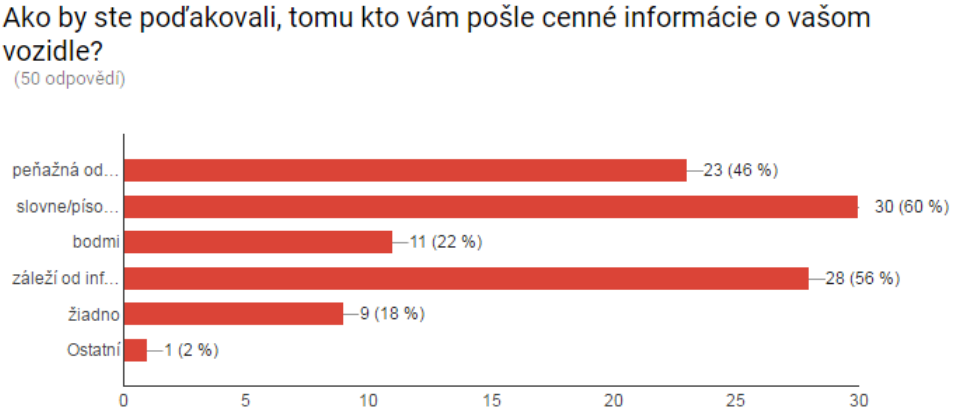

Obrázok 23: Otázka č.12



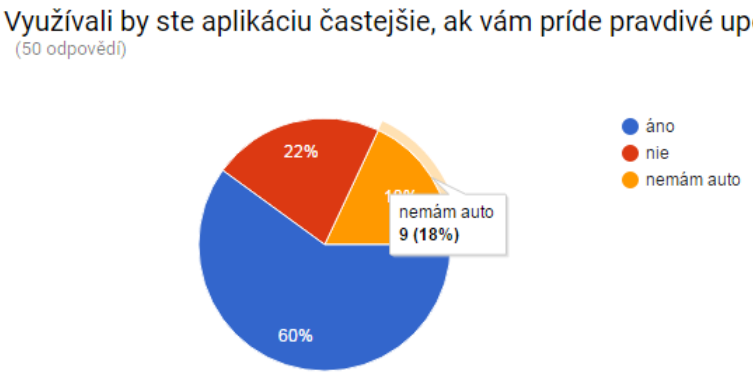

Obrázok 24: Otázka č.13




Obrázok 25: Otázka č.14




Obrázok 26: Otázka č.15

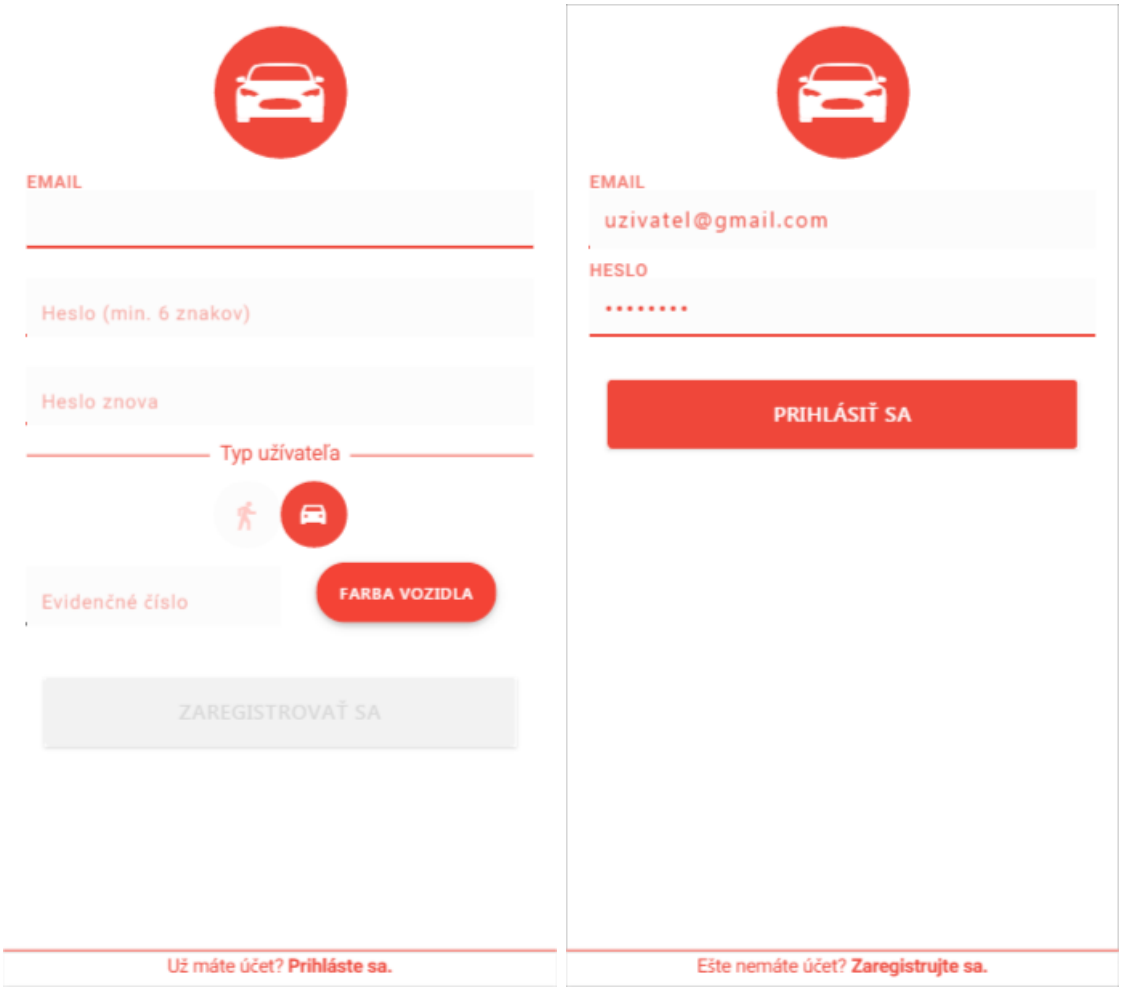



Obrázok 27: Otázka č.16

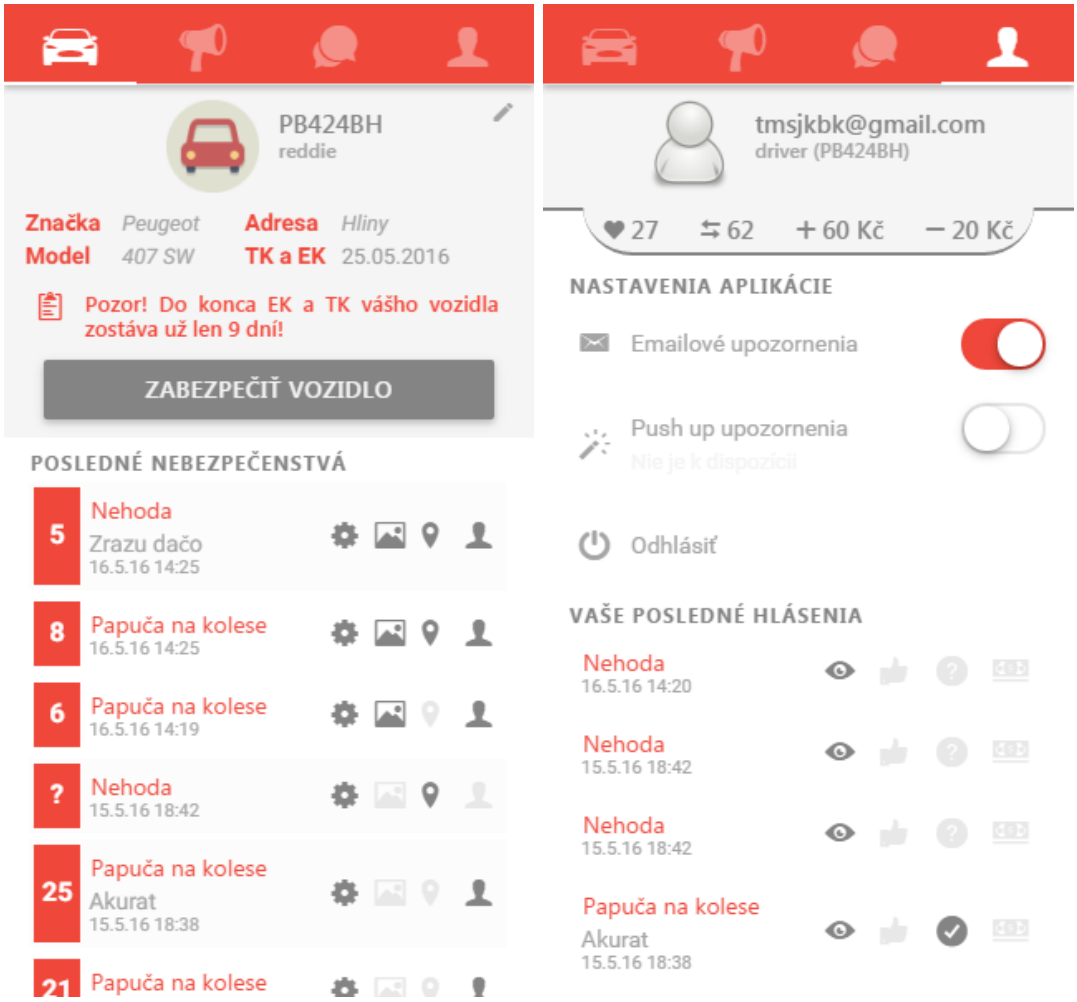

Obrázok 28: Otázka č.17


Obrázok 29: Otázka č.18

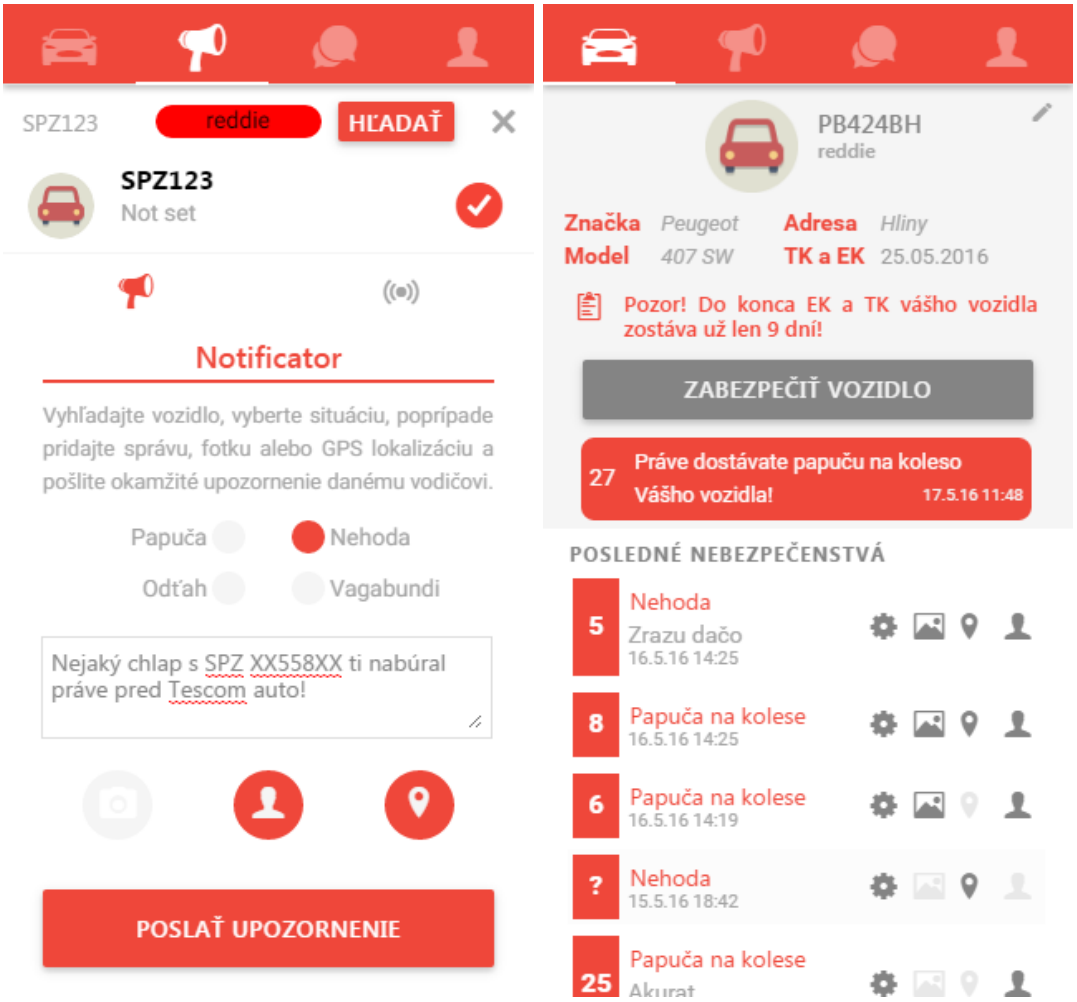
# B Ukážky obrazoviek



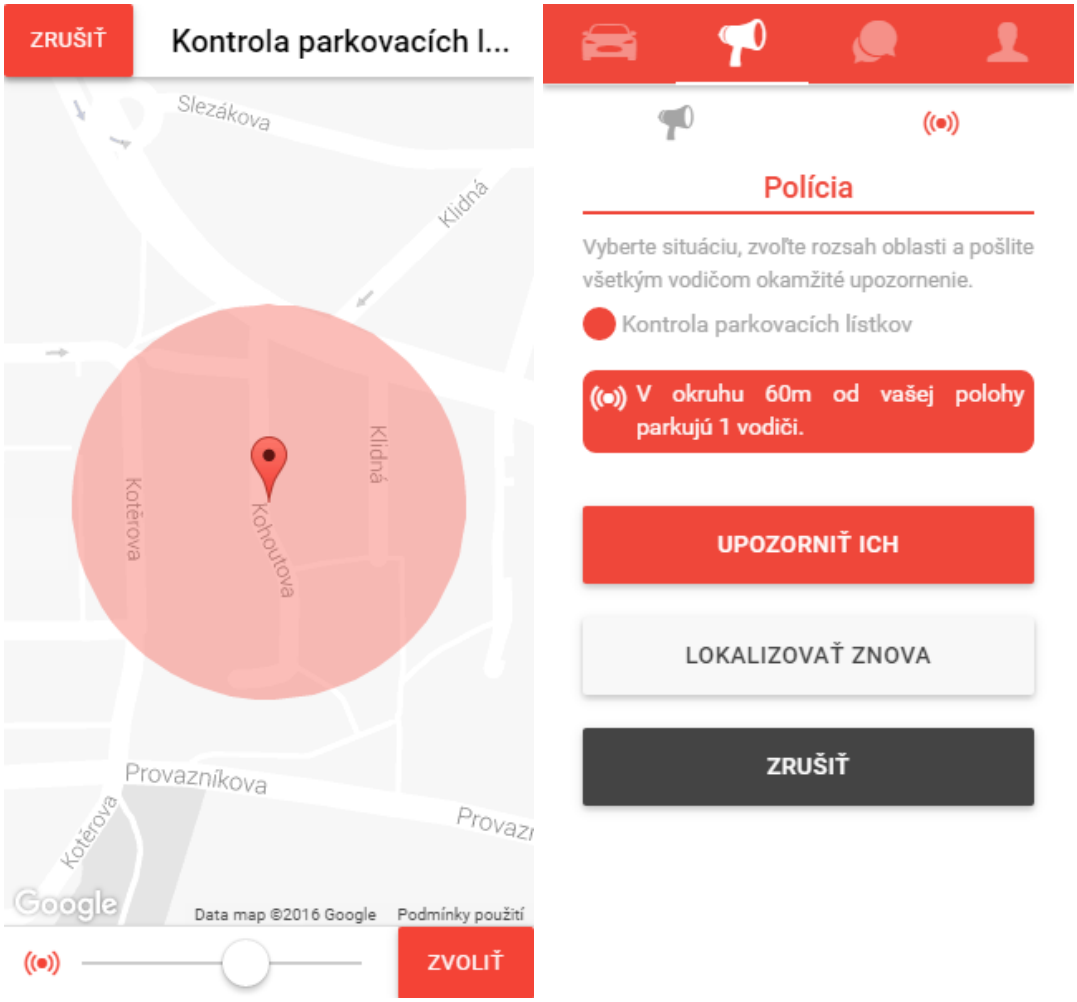

Obrázok 30: Registrácia a prihlásenie užívateľa



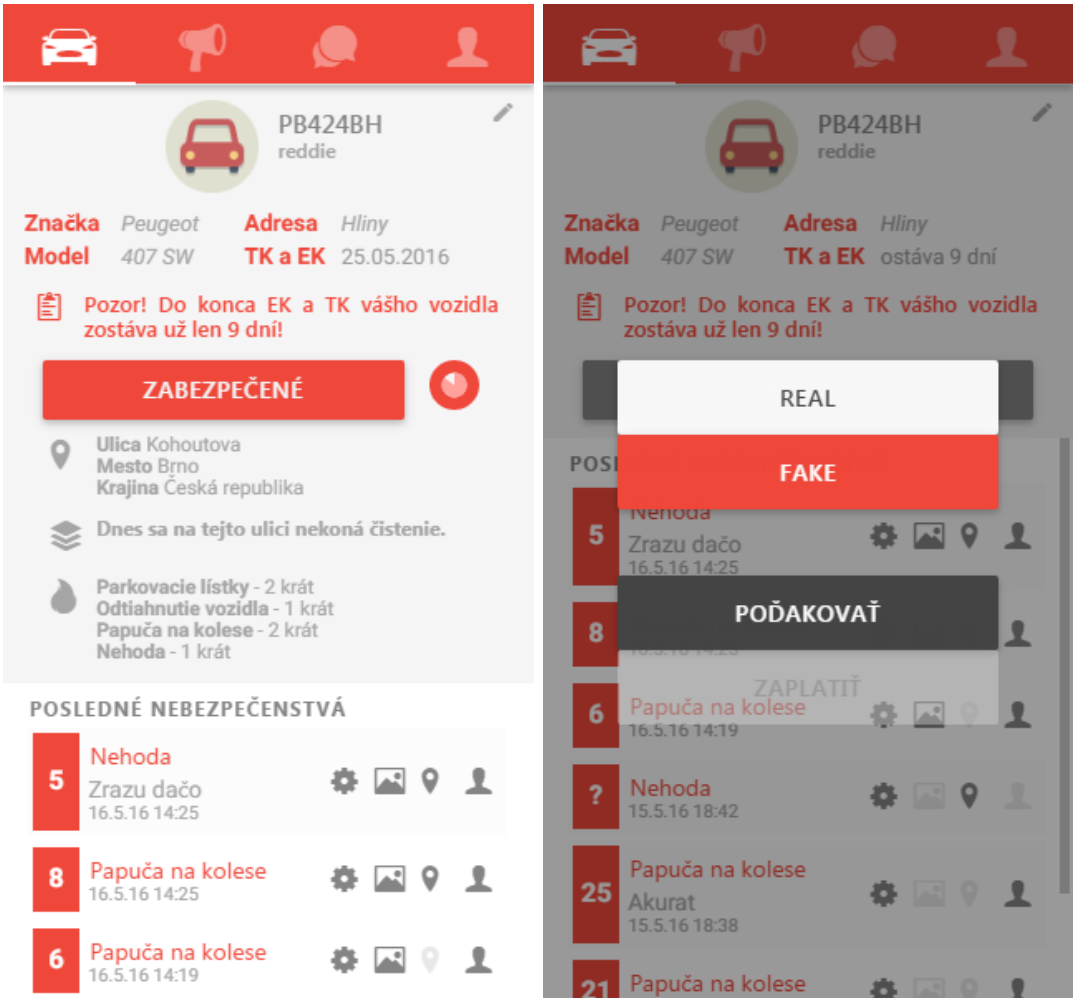

Obrázok 31: Záložka garáž a záložka profil užívateľa



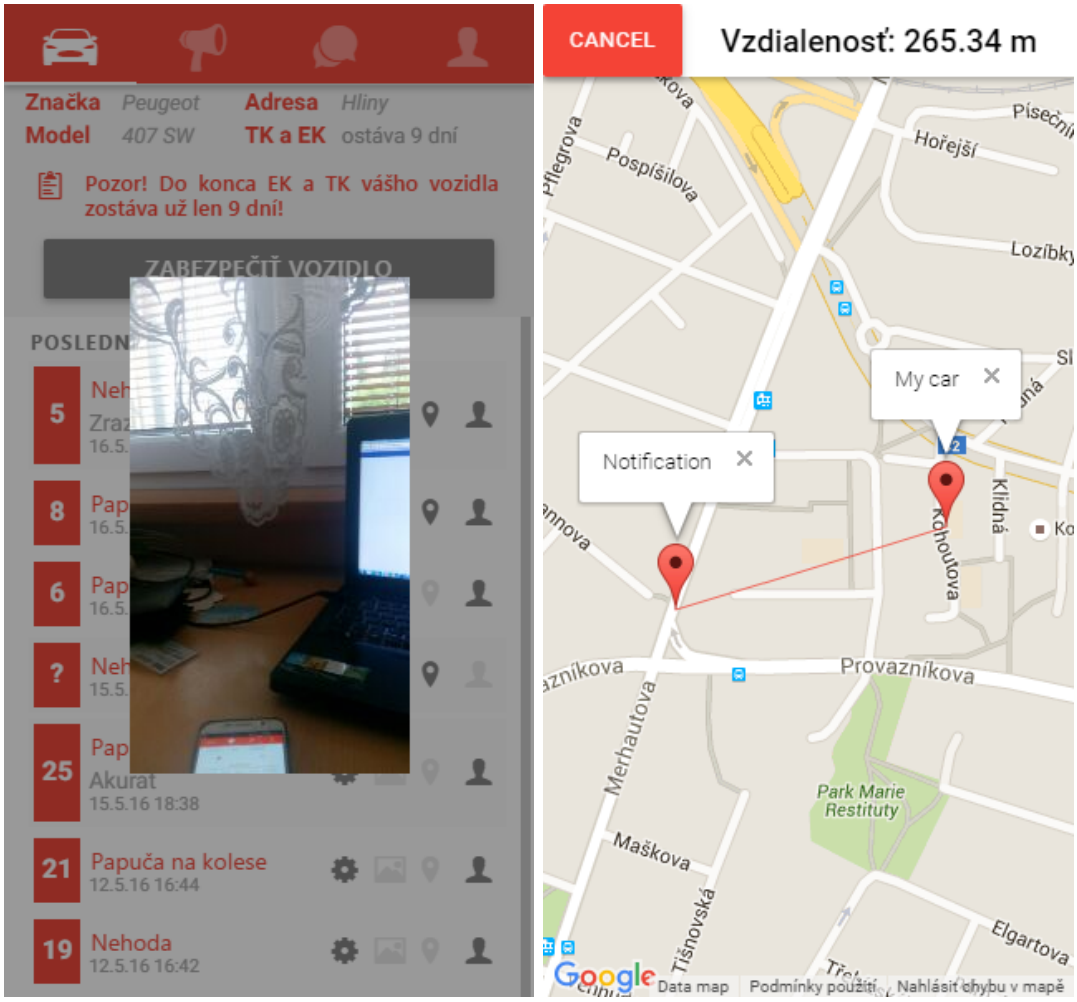

Obrázok 32: Poslanie a prijatie upozornenia



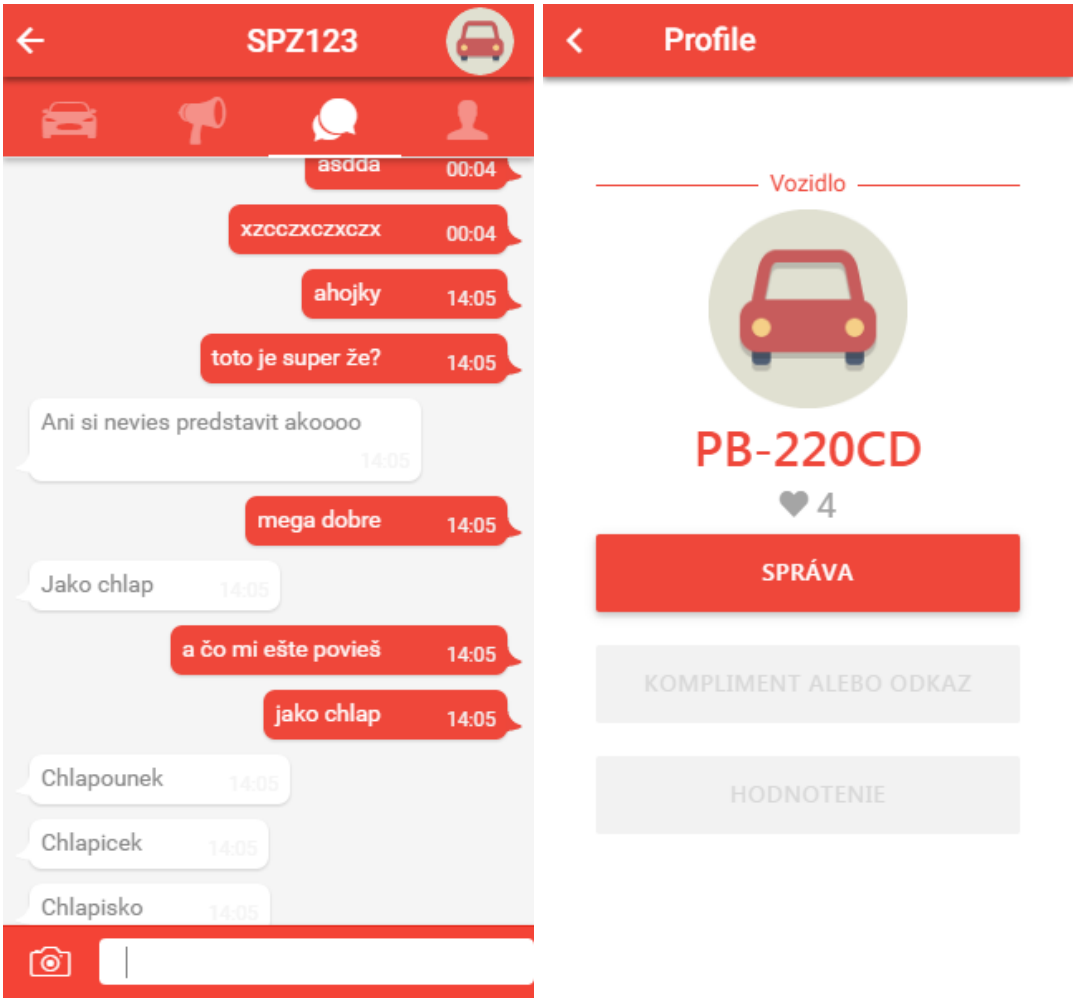

Obrázok 33: Voľba rozsahu pôsobenia upozornenia



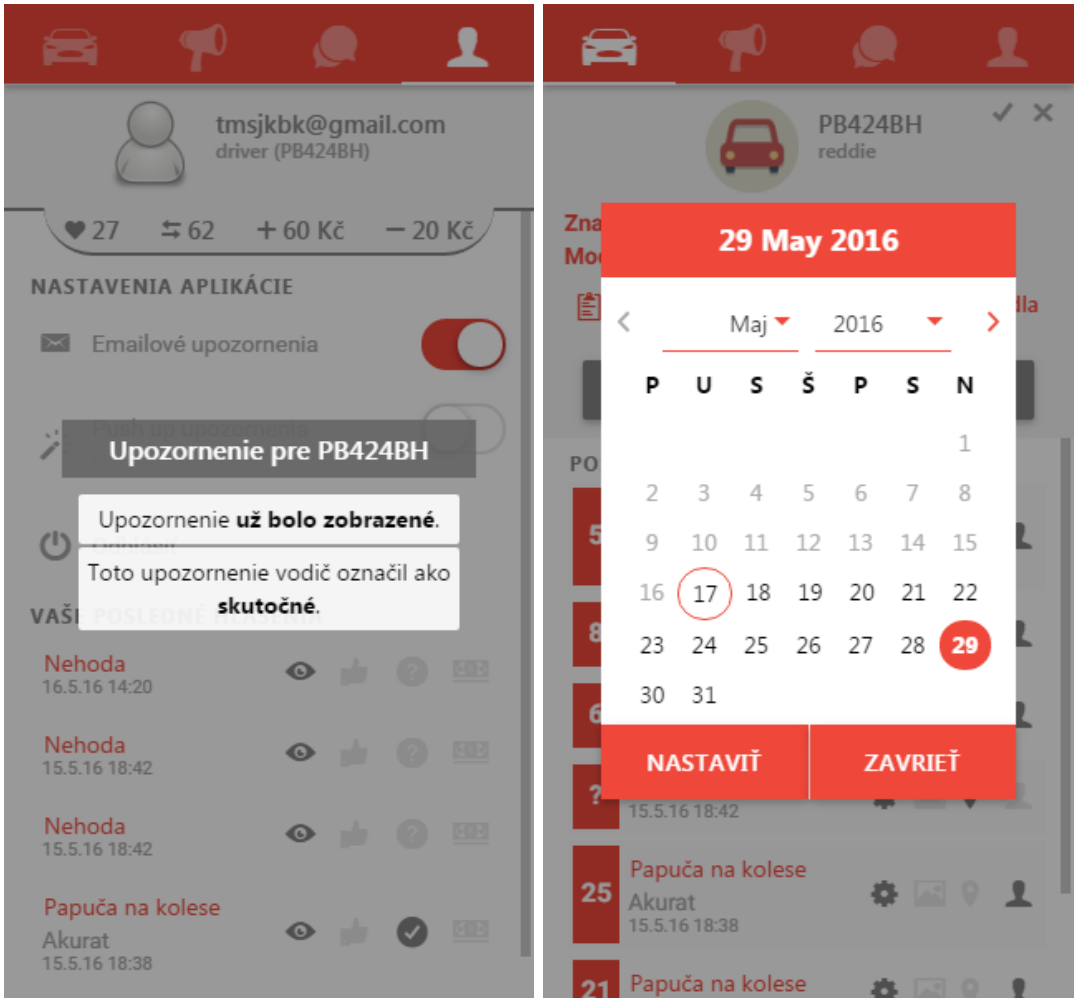

Obrázok 34: Zabezpečenie vozidla a hodnotenie upozornenia




Obrázok 35: Detaily prijatého upozornenia




Obrázok 36: Otvorená konverzácia a profil vozidla




Obrázok 37: Detail odoslaného upozornenia a kalendár na voľbu termínov TK a EK

# C CD s aplikáciou

Súčasťou tejto práce je aj CD nosič so zdrojovými kódmi aplikácie a postupom na jej spustenie a vyskúšanie.