VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ APLIKACE PRO SLEDOVÁNÍ POHYBU VOZIDEL

WEB APPLICATION FOR TRACKING OF VEHICLE MOVEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ONDREJ DUBAJ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21087/2017/xdubaj00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Dubaj Ondrej**
Obor: Informační technologie
Téma: **Webová aplikace pro sledování pohybu vozidel**
Web Application for Tracking of Vehicle Movement
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby webových a mobilních aplikací. Dále prostudujte možnost práce s GPS daty.
2. Analyzujte požadavky na aplikaci pro sledování vozidel, kde její mobilní část bude zodpovědná za odesílání GPS dat na server. Data budou zpracována webovou aplikací, která bude také umožňovat plánování tras, jejich vyhodnocení (čas jízdy, zpoždění atd.) a přiřazování jednotlivých řidičů k trasám.
3. Navrhněte aplikaci splňující výše uvedené požadavky. Návrh konzultujte s vedoucím.
4. Navrženou aplikaci implementujte a ověřte její funkčnost.
5. Zhodnoťte dosažené výsledky a navrhněte další možné pokračování tohoto projektu.

Literatura:

- Ujbányai, M.: Programujeme pro Android. Grada Publishing, 2012, ISBN 978-80-247-3995-3.
- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá štúdiom, návrhom, implementáciou a testovaním systému pre monitorovanie pohybu vozidiel. Systém je zložený z webovej a mobilnej aplikácie, kde webová aplikácia zabezpečuje plánovanie a interpretáciu údajov o trasách a mobilná aplikácia, ktorá je vyvinutá pre platformu Android, zabezpečuje zber dát a ich odosielanie na server. Systém dokáže zaznamenávať presné príchody a odchody vozidiel na kontrolné body, ich zdržania oproti plánovaným časom a prejdenú vzdialenosť. Back-end webovej aplikácie je implementovaný pomocou PHP frameworku Laravel a front-end pomocou CSS frameworku Bootstrap 4 a jeho šablóny Now UI Kit. Pre implementáciu mobilnej aplikácie bolo využité prostredie Android Studio SDK spolu s jazykom Java a značkovacím jazykom XML.

Abstract

The aim of this thesis is study, design, implementation and testing of vehicle movement monitoring system. The system consists of a web and a mobile application, where the web apllication provides route planning and data interpretation, and the mobile application, which is developed for Android, secures data collection and its sending to server. The system is able to record accurate arrivals and departures of vehicles at control points, their delays compared to planned times and the travelled distance. The back-end of web application is implemented using the Laravel PHP framework and front-end using the Bootstrap 4 CSS framework and its Now UI Kit template. The Android Studio SDK was used to implement the mobile application, along with Java and XML markup language.

Klíčová slova

GPS, webová aplikácia, mobilná aplikácia, sledovanie polohy, Android, Laravel, automatické zaznamenávanie časov a vzdialeností, interpretácia zdržaní

Keywords

GPS, web application, mobile application, position tracking, Android, Laravel, automatic recording of time and distance, interpretation of delays

Citace

DUBAJ, Ondrej. *Webová aplikace pro sledování pohybu vozidel.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Webová aplikace pro sledování pohybu vozidel

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondrej Dubaj
6. května 2018

Poděkování

Rád by som sa poďakoval pánovi Ing. Vladimírovi Batíkovi PhD. za vedenie a odbornú pomoc pri riešení tejto bakalárskej práce. Taktiež by som chcel prejaviť vďačnosť mojej rodine, priateľom a priateľke za zhovievavosť a podporu počas celého štúdia.

Obsah

Kapitola 1

Úvod

V súčastnej dobe prevláda potreba lokalizácie činností spojených s pohybom ľudí a vozidiel po Zemskom povrchu. Využíva sa na to najmä Global Positioning System (GPS), ktorý za pomoci družíc dokáže zaznamenávať polohu akéhokoľvek zariadenia. Primárne bol určený pre vojenské účely, avšak s rozvojom dopravy sa usídlil aj tejto oblasti a stal sa jej nevyhnutnou zložkou. Prvotne sa využíval najmä pre navigáciu a orientáciu leteckej dopravy, avšak momentálne sa rozšíril aj v nákladnej a osobnej automobilovej doprave.

Existuje mnoho systémov a zariadení pre monitorovanie pohybu akýchkoľvek objektov. Najmä systémy pre monitorovanie pohybu nákladnej dopravy sú značne rozvinuté a takmer každá spoločnosť zaoberajúca sa rozvozom osôb alebo tovaru disponuje takýmto systémom. Potreba týchto spoločností je najmä zaznamenávanie príchodov a odchodov na dané miesta rozvozu, vyhodnocovanie zdržaní a následnú kontrolu svojich zamestnancov.

Cieľom tejto práce je návrh a implementácia práve takéhoto systému, ktorý by mohol slúžiť menším podnikateľom ako pomôcka pre zaznamenávanie vyššie spomenutých informácii a tak by sa zamestnanci mohli vyhnúť pracnému a ručnému zaznamenávaniu údajov, ktoré niesu prístupné v reálnom čase a na jednom mieste. Taktiež je tu menšia šanca chyby z dôvodu ľudského omylu, nakoľko kvalitný softvér by mal fungovať spoľahlivo a bezchybne.

V tejto práci bude v jednotlivých kapitolách predstavený vývoj aplikácie tohto typu za pomoci webových a mobilných technológií. Pre lepší prehľad je práca rozdelená na teoretickú a praktickú časť, kde v teoretickej časti (nasledujúce tri kapitoly) sa venujeme všeobecnému popisu a predstaveniu využitých technológii, a v praktickej časti budú popísané problémy spojené s vývojom konkrétneho systému. V kapitole 2 budú popísané základné princípy webových aplikácií, technológie využívané pre ich tvorbu a podrobnejšie popísané technológie využité pre implementáciu tohto systému. V kapitole 3 sú popísané najrozšírenejšie mobilné operačné systémy využívané v inteligentných telefónoch a technológie využívané pre ich vývoj. Predstavenie princípov fungovania GPS s vysvetlením funkcionality na príklade bude uvedené v kapitole 4. V kapitole 5 bude zhodnotená analýza požiadaviek, na ktorú bude nadväzovať kapitola 6 v ktorej je uvedený návrh systému. Kapitola 7 popisuje implementáciu konkrétnych častí systému. Testovaniu jednotlivých častí, ale aj systému ako celku, je venovaná kapitola 8. Možnosti rozšírenia systému a záverečné zhrnutie sú popísané v kapitolách 9 a 10.

Kapitola 2

Webové aplikácie

Webové aplikácie, známe najmä ako webové stránky, sú v súčasnej dobe pravdepodobne najviac rozšírený reklamný, informačný a vzdelávací prostriedok. V posledných rokoch zaznamenali veľký rozmach, či už z technologickej stránky alebo v každodennom používaní. Využívajú sa prakticky vo všetkých sférach bežného života, ako napríklad nákupy, ponuka služieb, vyhľadávanie informácii, plánovanie udalostí a podobne.

Pod pojmom webovej stránky alebo aplikácie si bežný užívateľ predstaví interaktívne a oku lahodiace dokumenty zložené z textu, obrázkov a rôznych animácii, na ktorých nájde vždy všetky potrebné informácie. Nie nadarmo sa vraví, že dnes sú všetky informácie na internete.

Vývojári webových aplikácii si ale pod týmto pojmom predstavia súhrn navzájom prepojených HTML dokumentov dostupných pomocou protokolu HTTP, ktorý slúži na prenos informácii z webového servera do klientskej aplikácie známej ako internetový prehliadač. Všetky tieto webové servery spolu tvoria **World Wide Web**[19]. Ku každej webovej stránke je možné pristúpiť pomocou IPv4 alebo IPv6 adresy zadanej do internetového prehliadača, avšak s vývojom DNS (Domain Name System) a ich A a AAA záznamov sa k nim pristupuje pomocou doménových mien.

S rozšírením inteligentných telefónov začali byť webové aplikácie prispôsobované pre menšie obrazovky. Týmto problémom sa zaoberajú dizajnéri, ktorí sa snažia o čo najefektívnejšie rozloženie HTML elementov na obrazovke akejkoľvek veľkosti. Takúto vlastnosť aplikácií nazývame **responzivita**[4].

Typy webových aplikácii[16]:

- **Statické** - sú zložené zo statických HTML dokumentov doplnených o CSS a Javascript technológie. Užívateľovi sú zo serveru doručené presne tak ako sú uložené na serveri. Zobrazuje každému užívateľovi rovnaký obsah, avšak môžu podporovať aj multijazyčnosť. Sú primárne určené iba na prezeranie, čo znamená, že neposkytujú užívateľovi takmer žiadnu interakciu. Majú zväčša informačný charakter.

- **Dynamické** - sú to statické webové stránky doplnené o skriptovacie technológie umožňujúce interaktívny prístup užívateľa. Ich obsah je vygenerovaný aktuálnymi informáciami pre každé individuálne zobrazenie, čo znamená, že obsah sa mení v závislosti na čase, užívateľovi a kontexte. Ďalej sa budeme zaoberať dynamickými webovými aplikáciami.

2.1 História

Vznik prvých webových stránok sa datuje do marca roku 1989, kedy vedci z inštitútu *CERN* vo Švajčiarsku vytvorili dokument s názvom *Manažment Informácii: Návrh*[17]. V tomto dokumente sa podľa informácii médií nič neobjavilo. Postupom času ale tento dokument zaznamenal zásadný obrat vo svete sprostredkovania informácii.

Spočiatku existovali iba statické webové aplikácie, ktoré sa postupom času a najmä v poslednom desaťročí vyvinuli v plne dynamické webové aplikácie.

2.2 Technológie

Pre vývoj webových aplikácii existuje veľké množstvo rôznych technológii. Vývoj môže byť rozdelený do dvoch základných oblastí (modelov). Prvá je na strane klienta (**client-side**) kde ide napríklad o technológie AJAX, JavaScript, Microsoft Silverlight, Adobe Flash, HTML, CSS, a ďalšie. Na strane servera (**server-side**) sú to technológie napríklad ako ASP.NET, C#, .NET, PHP, Java, Perl a podobne[21][6].

2.2.1 Klientská časť aplikácie

Pre zobrazenie obsahu webovej aplikácie na strane klienta (užívateľa) je zodpovedná klientská aplikácia - internetový prehliadač. Využívajú sa najmä značkovacie jazyky, kaskádové štýly a Javascript. Tieto technológie budú popísané v ďalších podkapitolách.

HTML

Najrozšírenejším značkovacím jazykom používaných vo webových technológiách je HTML. Je aplikáciou skôr vyvinutého univerzálneho značkovacieho jazyka SGML. Jeho vývoj je ovplyvnený najmä vývojom internetových prehliadačov[26].

HTML je charakterizovaný množinou značiek a ich vlastností. Jednotlivé značky je možné do seba vnorovať, čím sa vytvára hierarchická štruktúra obsahu stránky.

Medzi značky sa uzatvárajú časti textu dokumentu a tým sa určuje význam obsiahnutého textu. Značky môžeme rozdeliť na[9]:

- **Párové** - majú obsah, koncová značka je zhodná s počiatočnou, ale má pred názvom znak lomítko, napríklad * Textové pole *
- **Nepárové** - nemajú obsah, nepoužívajú žiadnu koncovú značku, napríklad *<hr>*

Ďalším možným rozdelením je na základe štruktúrovania obsahu, tam rozoznávame[22]:

- **Riadkové elementy** - neovplyvňujú zalomenie textu
- **Blokové elementy** - spôsobujú zalomenie riadku (odsadenie, vynechanie miesta, ...)

Poslednou verziou jazyka HTML je HTML5.

CSS

Cascading Style Sheets, inak Kaskádové štýly je jazyk pre popis spôsobu zobrazenia elementov na stránkach napísaných značkovacími jazykmi. Cieľom je umožniť návrhárom oddeliť vzhľad dokumentu od jeho štruktúry a obsahu[30].

Syntax kaskádových štýlov pozostáva z niekoľkých pravidiel. Každé pravidlo obsahuje selektor a blok deklarácii. Každý blok deklarácii obsahuje deklaráciu oddelenú bodkočiarkami a každá deklarácia sa skladá z identifikátoru vlastnosti a hodnoty oddelenými dvojbodkou.

Typy selektorov[5]:

- HTML element
- Identifikátor
- Trieda

Poslednou verziou kaskádových štýlov je CSS3.

JavaScript

Javascript je multiplatformný objektovo-orientovaný skriptovací jazyk, ktorý bol vyvinutý Brendanom Eichom v jeho tehdajšej spoločnosti Netscape. Používa sa ako interpretovaný programovací jazyk pre webové stránky, kde je vkladaný priamo do HTML kódu stránky[24]. Nevýhodou je, že jazyk JavaScript je slabo typovaný, čo môže viesť ku skrytým chybám zo strany programátorov. V súčastnej dobe sa do popredia dostáva jeho typovaná verzia **TypeScript**[31], ktorá je v provom kroku kompilovaná do JavaScriptu a až následne do jednoduchších jazykov.

Narozdiel od ostatných interpretovaných programovacích jazykov, ktoré spúšťajú svoj kód na strane serveru, je kód Javascriptu spúšťaný až po stiahnutí stránky zo serveru priamo v internetovom prehliadači. Jeho najvýznamnejšie využitie nachádzame pri dynamických zmenách obsahu stránky bez potreby posielať ďalšie požiadavky na server, čím docielime zrýchlenie aplikácie a ušetríme užívateľa nepríjemného "prebliknutia"(znovustiahnuntia a znovunačítania) prehliadanej stránky. Najrozšírenejšou knižnicou pre tvorbu webových aplikácii je **JQuery**[13].

2.2.2 Serverová časť aplikácie

Serverová časť aplikácie slúži na generovanie obsahu dynamických webových aplikácii. Užívateľ pomocou svojho internetového prehliadača odošle HTTP požiadavku na server. Daná požiadavka je spracovaná a klientovi je odoslaná odpoveď vo forme HTML kódu.

PHP

PHP (Personal Home Page Tools) je interpretovaný multiplatformný programovací jazyk vyvinutý pre generovanie dynamického obsahu webových aplikácii. Zdrojový kód sa vkladá priamo do HTML kódu vo forme príkazov ohraničených značkami *<?php* a *?>*.

PHP nepatrí medzi silno typované programovacie jazyky a premenné neni nutné dopredu deklarovať. Ďalším špecifikom tohto jazyka je flexibilita premenných v tvare polí, kde indexom môžu byť čísla ale aj reťazce. Veľkosť polí je dynamická, takže nie je potrebné dopredu deklarovať veľkosť poľa[27].

Webový server musí obsahovať modul pre spracovanie PHP, ktorý je potom spustený priamo v kontexte procesu webového serveru. Súbor s príponou *.php* je analyzovaný interprétom jazyka PHP a ak obsahuje značky ohraničujúce skript, interpret vykoná dané príkazy a výsledný HTML súbor vráti webovému serveru, ktorý ho preposiela klientovi. Ak niesu v danom súbore nájdené žiadne značky, interpret vráti skript v nezmenenej podobe.

MySQL

Každá webová aplikácia využívajúca skriptovací jazyk PHP väčšinou obsahuje aj databázový systém. Databázy slúžia na perzistentné uchovávanie dát. U webových aplikáciach sú najviac rozšírené relačné databázy MySQL, ktoré sú multiplatformné a pre svoje dotazy využívajú jazyk SQL[23].

Pre pripojenie na databázový systém sa využíva jazyk PHP a pre chod celého systému je potrebný aj server (napríklad **Apache**)[33]. Najrozšírenejšou variantou technológii pre chod webových aplikácii je LAMP (Linux), WAMP (Windows) alebo XAMP (Mac), ktoré spájajú kombináciu Apache, Mysql a PHP.

Laravel

Laravel je aplikačný open-source PHP framework, poskytovaný pod licenciou MIT. Je jeden z najrozšírenejších frameworkov vyvinutých pre uľahčenie tvorby malých, stredne veľkých ale aj veľkých webových aplikácii. Laravel využíva architektúru MVC (Model-View-Controller)[11], kde *Model* reprezentuje databázovú časť aplikácie, *View* klientske rozhranie a *Controller* riadiacu logiku celej aplikácie. Prvý krát bol popísaný v **SmallTalk 80** a stal sa z neho najpoužívanejší programátorský model[10].

Laravel je optimalizovaný pre reálny svet, čo znamená, že obsahuje často využívané vopred predprogramované metódy. V porovnaní s inými PHP frameworkmi pre tvorbu webových aplikácii, ako napríklad *CodeIgniter*, *Zend*, *Symfony2* alebo *Yii*, je Laravel jeden z najmladších avšak najrýchlejšie sa rozmáhajúcich. Pomocou rôznych nástrojov pre sledovanie záujmu je možné zistiť, že získava na popularite najmä vďaka jednoduchosti a možnosti upraviť si samotný framework podľa svojich potrieb, čo iné frameworky nemusia umožňovať. Svedčia o tom aj krivky z vývojárskych stránok ako **stackoverflow.com** alebo **github.com**, kde sa Laravel v počte vyhľadávaní alebo sledovaní umiestňuje na popredných pozíciách. Tým, že sa jedná o voľne šíriteľný softvér, počet prispievateľov je veľký a tak, ako napríklad operačný systém Linux, disponuje funkcionalitou presne mierenou pre potreby vývojára.

Využíva a opiera sa o podporné systémy ako *Symfony*, *Composer*, *Equolent* či *Blade*. Taktiež disponuje otvorenými postupmi inšpirovanými technológiami ako napríklad *Ruby on Rails*, *ASP.NET* a podobne. Rovnako ponúka vývojárovi jednoduchý spôsob unit-testovania určitých častí kódu pomocou nástroja **Artisan**.

Základnú funkcionalitu, ktorú Laravel podporuje je[14]:

- **Autentizácia** - možnosť vygenerovať si kontrolu prístupu užívateľov
- **Routovanie** - správa a smerovanie požiadaviek
- **Databáza** - možnosť automatickej migrácie databázových tabuliek
- **Mail** - posielanie mailov s prílohami
- **Sessions** - dočasné súbory na uloženie dát webovej aplikácie
- **Caching** - kešovanie používaných dát

Adresárová štruktúra Východiskovým bodom celého frameworku je koreňový adresár, ktorý obsahuje množstvo ďalších adresárov. Následne si uvedieme tie najpodstatnejšie z nich:

- **/app** - obsahuje kód jadra aplikácie, nájdeme v ňom ďalšie adresáre ako */Console*, */Http*, */Providers*, */Controllers* a väčšinou súbory (triedy) reprezentujúce dátové modely.
- **/bootstrap** - obsahuje sadu súborov, ktoré zabezpečujú automatické načítanie
- **/config** - obsahuje konfiguračné súbory ako napríklad pripojenie k databáze, odosielanie emailov, nastavenie kešovania a autorizačné pravidlá
- **/database** - obsahuje migrácie (súbory reprezentujúce jednotlivé tabuľky relačnej databázy)
- **/routes** - obsahuje súbory smerovania medzi jednotlivými časťami aplikácie, spája logické prvky aplikácie s užívateľským rozhraním
- **/public** - disponuje obsahom aplikácie, ako napríklad obrázky, CSS, JavaScript a podobne
- **/resources** - obsahuje súbory užívateľského rozhrania, šablóny a súbory lokalizácie
- **/storage** - obsahuje cache pamäť súborov, relácie a skompilované šablóny typu Blade
- **/tests** - obsahuje zautomatizované jednotkové testy
- **/vendor** - obsahuje väzby na Composer

Základný cyklus Všetky žiadosti prijímané od klienta sú zasielané cez súbor *public/index.php*. Potom, čo klientská požiadavka vstúpi do súboru *index.php*, je načítaný súbor *bootstrap/start.*php. Tento súbor vytvorí nový aplikačný objekt Laravelu, ktorý slúži ako *IoC* (Inversion of Control) kontajner. IoC kontajner je určený najmä na riadenie závislostí jednotlivých tried. *Dependency injection* je metóda odstránenia pevne implementovaných častí kódu a ich nahradenia až v čase behu programu, čo umožňuje väčšiu flexibilitu[12].

Po vytvorení aplikačného objektu sa nastaví niekoľko ciest a prebehne detekcia behového prostredia. Potom je zavolaný interný bootstrap skript, ktorý nastavuje konfiguračné nastavenia ako časové pásmo, chybové hlásenia a podobne. Rovnako registruje všetkých poskytovateľov služieb nakonfigurovaných pre našu aplikáciu. Potom, čo sú všetky služby registrované, sú načítané súbory z *app/start*. Nakoniec je načítaný súbor *app/routes.php*, po ktorom je objekt so žiadosťou zaslaný aplikácii.

Kapitola 3

Mobilné aplikácie

Inteligentné telefóny, inak smartfóny, sú neodmysliteľnou súčasťou každodenného života väčšiny ľudí. Ponúkajú nám obrovské možnosti získavania akýchkoľvek informácii a aj vďaka nim prebieha digitalizácia v takmer všetkých odvetviach bežného života.

Medzi najväčšie výhody inteligentých telefónov patria:

- **Mobilita** - funkcionalitu zariadení môžeme využívať kdekoľvek a kedykoľvek
- **Veľkosť** - sú malé a prenosné
- **Multifunkčnosť** - poskytujú radu funkcionalít, napríklad prístroj na telefonovanie, navigácia, fotoaparát

Mobilné aplikácie sú prostriedkom užívateľa pre riadenie inteligentného telefónu. Tvoria rozhranie medzi užívateľom a samotným hardvérom. Jednoducho povedané, je to program bežiaci na mobilnom zariadení. Pre každú funkciu telefónu slúži minimálne jedna aplikácia.

Nakoľko inteligentný telefón sa výkonnostne nedokáže vyrovnať stolovému počítaču, ako napríklad obmedzenou kapacitou batérie, nižším výkonom procesora alebo menšou kapacitou operačnej pamäte, je potrebné mobilné aplikácie prispôsobiť hardvéru aký je k dispozícii. Bolo by ideálne aby mala aplikácia čo najmenšiu energetickú spotrebu batérie, ktorá vyplýva z dĺžky zaťaženia procesora. Aplikácia by z toho dôvodu mala byť rýchla, pretože plne zaťažený procesor čerpá výrazne väčšie množstvo energie ako nezaťažený. V prípade, že sa zaťaženiu procesora nevieme vyhnúť, mal by byť zaťažený čo najkratšiu dobu. Rovnakú stratégiu pri tvorbe mobilných aplikácii je potrebné uplatniť aj vzhľadom na vyťaženosť operačnej pamäte. Z tohto dôvodu by mali aplikácie vyžadovať a zaberať malé množstvo pamäte. Takto navrhnuté aplikácie sú ideálne pre prácu v teréne, kde nie je k dispozícii stolný počítač. Avšak pri práci s náročnými výpočtami prebieha iba aproximácia výsledkov už z vyššie spomenutých dôvodov nedostatočnosti výkonu. Preto sú takéto zariadenia ideálne najmä pre zber dát a ich presné vyhodnocovanie je lepšie vykonávať na výkonnejších strojoch.

3.1 História

Prvé telefóny boli vytvorené v roku 1940, kde za základe bunkovej technológie boli určené najmä pre používanie v autách záchranných zložiek a v taxíkoch. Skutočný začiatok komerčného využívania telefónnych zariadení sa ale datuje do roku 1983, kedy boli verejnosti predstavené mobily prvej generácie (1G) založené na analógovej technológii. Po tomto historickom miľníku začal skutočný rozkvet mobilných telefónov, kedy už v roku 1990 boli

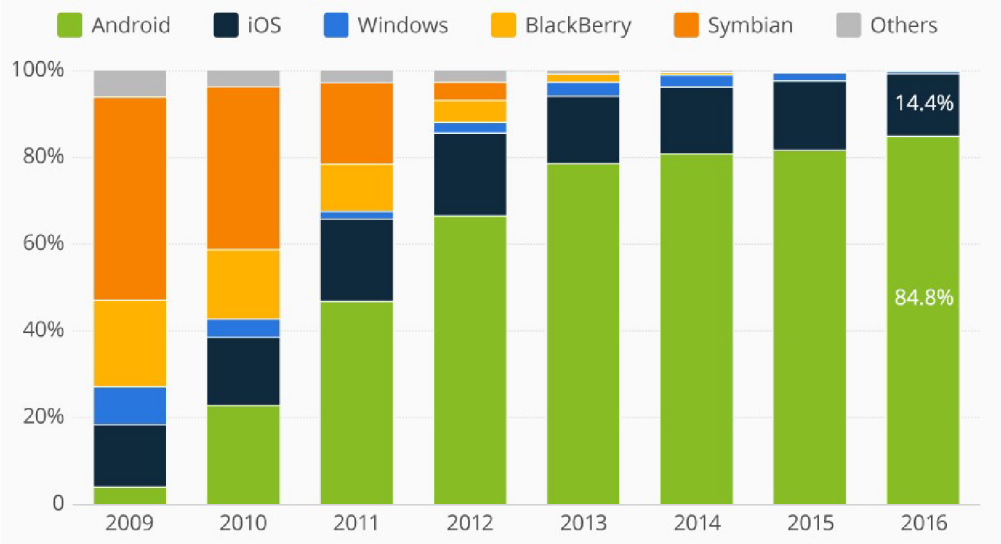
na trhu telefóny druhej generácie (2G) využívajúce digitálnu technológiu. Veľmi skoro po vzniku 2G technológie sa vyvinula technológia tretej generácie (3G), ktorá okrem prenosu zvuku umožňuje prenos aj iného typu údajov ako napríklad emaily, správy a iné. Táto technológia je používaná dodnes. V súčasnosti prichádza do popredia technológia štvrtej generácie (4G), ktorej maximálny rozmach nás ešte len čaká. Zabezpečuje rýchlejší prenos dát a ponúka nové funkcie[8].

Rozmach mobilných aplikácii prebiehal súčasne s telefónmi, avšak prelom nastal v roku 2007, keď spoločnosť *Apple* ponúkla trhu prvý telefón s dotykovým displejom. Od tohto dátumu začal masívny vývoj aplikácií všetkého druhu.

3.2 Operačné systémy

V súčasnosti existuje viacero operačných systémov pre mobilné zariadenia, z ktorých najznámejšie sú **Android** vyvíjaný spoločnosťou *Google*, **iOS** od spoločnosti *Apple*, **Windows Phone** od *Microsoftu* a **BlackBerry OS** od spoločnosti *Research in Motion (RIM)*[25]. Detailnejší popis jednotlivých operačných systémov bude v nasledujúcich kapitolách.

Vedúcu pozíciu v percentuálnom zastúpení mobilných operačných systémov vo svete má Android. Práve z tohto dôvodu sme sa rozhodli mobilnú časť našej aplikácie vyvíjať pre túto platformu, čo bude mať za následok možnosť širokého využitia systému. Ako je vidieť na priloženom obrázku nižšie, tento systém by v prípade potreby mohlo využívať takmer 85 percent užívateľov sveta bez potreby zmeny operačného systému alebo zakúpenia si iného zariadenia.



Obrázek 3.1: Zastúpenie mobilných operačných systémov na svetovom trhu

3.2.1 Android

Android je operačný systém založený na linuxovom jadre, ktorý je dostupný ako otvorený softvér (open-source)[28]. Je používaný na inteligentných telefónoch, tabletoch, televíziach a ďalších zariadenach. V súčastnosti má vo svete najväčšie zastúpenie medzi operačnými

systémami. Z dôvodu, že obsahuje linuxové jadro je možný jeho beh na širokom spektre fyzických zariadení. Z tohto dôvodu si výrobcovia hardvéru dokážu systém ľubovolene prispôsobiť podla svojich potrieb.

Vývoj Android aplikácii prebieha za pomoci multiplatformných jazykov **Java**[1] a momentálne sa do popredia dostáva aj jazyk **Kotlin**. Najpoužívanejším prostredím pre vývoj je **Android SDK**, ktorý obsahuje sadu knižníc, emulátor, dokumentáciu a ukážky kódov. Je dostupný ako voľne šíriteľný softvér pre Linux, Mac aj Windows.

3.2.2 iOS

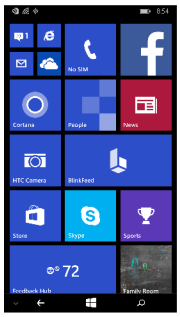
iOS je operačný systém určený pre zariadenie vyvíjané spoločnosťou Apple. Je odľahčenou verziou OS X a pôvodne bol určený iba pre zariadenia iPhone, avšak jeho výhody medzi ktoré patrí minimálne spoľahlivosť, intuitívnosť a jednoduchosť ovládania ho rozšírili na ďalšie zariadenia ako iPad, iPod, Apple TV a podobne[29].

Ako už bolo spomenuté, bol to prvý operačný systém pre inteligentný telefón, z čoho vyplýva, že prvé mobilné aplikácie boli vyvinuté práve pre túto platformu. Jadro systému je založené na *Darwin OS* obsahujúci nízkoúrovňové funkcie, na ktorých je postavená väčšina ostatných technológii.

Pre vývoj iOS aplikácii sa využíva najmä **jazyk C** a jeho nadstavba **Objective-C** a taktiež jazyk **Swift**. Prostredie **iOS SDK** je voľne šíriteľné, avšak nevyhnutnosťou je operačný systém Mac OS X a počítač od spoločnosti Apple. Týmto sa ukazuje aj jedna z najväčších nevýhod vývoja aplikácii pre operačný systém iOS, ktorou je závislosť vývojárov na samotnej platforme.

3.2.3 Windows Phone

Windows Phone je operačný systém je najmladším predstaviteľom platforiem pre inteligentné telefóny. Je nasledovníkom staršej verzie systému Windows Mobile. Prvykrát bola verzia 7 predstavená v roku 2010 v USA a ponúkla nové užívateľské rozhranie s názvom *Metro*[32]. To ponúka domovskú obrazovku zloženú z dlaždíc, ktoré reprezentujú rôzne užívateľom zvolené aplikácie.



Obrázek 3.2: Ukážka užívateľského rozhrania operačného systému *Windows Phone*

V súčasnosti sú najrozšírenejšie verzie Windows Phone 7, Windows Phone 8 a Windows 10, ktorá je univerzálnym systémom a pre inteligentné telefóny sa využíva jeho upravená verzia *Windows 10 Mobile*. Pre vývoj aplikácii na platformu 7 sa využíva .NET framework obsahujúci základné funkcie pre vývoj všetkých Windows aplikácií vrátane tých deskopo-

vých. Windows Phone 8 a novšie obsahujú nový typ jadra a bola pre nich vyvinutá nová aplikačná architektúra. Tá podporuje vývoj aplikácii v jazykoch **C#**, **VB.NET**, **C/C++** alebo aj **HTML**. Nevýhodou je nekompabilita verzií 7 a 8[18].

Taktiež aj pre túto platformu je vyvinuté SDK obsahujúce hlavné vývojové prostredie, softvér pre vývoj grafického rozhrania a emulátor fyzických zariadení.

3.2.4 BlackBerry

BlackBerry je operačný systém známy najmä pre svoju podporu firemnej korešpondencie a synchronizáciu úloh, kalendára alebo kontaktov medzi prístrojom a účtom užívateľa. Je to poprietálna mobilná platforma podporujúca multitasking a viacero špecializovaných vstupných zariadení prispôsobené firmou *RIM* pre použitie s ich handheldami, ktoré predstavujú hardvérové zariadenia umožňujúce hlasové a dátové služby[3][2].

Pre vývoj aplikácii sa využívajú jazyky **C/C++**, **Java**, **HTML**, **CSS**, **Javacsript** a vývojové prostredie **BlackBerry 10 Native SDK**. Ako každé vývojové prostredie ponúka sadu knižníc a emulátor zariadenia.

Kapitola 4

GPS

Globálny polohový systém (GPS) je družicový systém pre určovanie polohy a času na zemskom povrchu a v priľahlom priestore. Je schopný poskytovať tieto údaje nezávisle na počasí a 24 hodín denne. Družice vysielajú signály, ktoré sú prijímané prijímačmi a spracované pre výskumné alebo navigačné účely. Presnosť stanovenia horizontálnej polohy sa pohybuje v rozmedzí od stoviek metrov až po niekoľko centimetrov. Všetky spôsoby využitia smerujú k získavaniu priestorových dát (polohy v čase), a to buď statických alebo dynamických.

GPS pôvodne vznikol ako vojenský projekt a bol využívaný najmä pre vojenské účely týkajúce sa orientácie a navigácie v známom i neznámom teréne. Začal sa rozvíjať už koncom 70. rokov 19. storočia, ale až v posledných rokoch sa dostal do povedomia širšej verejnosti a začal byť dostupný aj pre bežných užívateľov. Jeho vznik súvisí najmä s rozvojom vesmírneho programu, kedy sa prešlo od tradičného spôsobu orientácie a navigácie podľa bodov na Zemi k navigácií pomocou technológii. Vypustením prvých družíc monitorujúcich polohu a povrch Zeme začala revolúcia v oblasti určovania polohy a navigácie. V súčasnosti má mimo armádnych a vojenských zložiek milióny bežných užívateľov[15].

Presnosť určovania polohy závisí najmä od prijímačov. Medzi najpresnejšie patria technické zariadenia určené pre vojenské alebo geodetické účely. Dosahujú presnosť až niekoľko milimetrov. Pre širokú verejnosť nie je takáto presnosť potrebná a ani finančne prijateľná. Momentálne sú najrozšírenejšie navigácie do automobilov, hodinky určené pre športové účely, či v poslednej dobe najviac rozmáhajúce sa inteligentné telefóny. Tieto bežným užívateľom dostupné zariadenia sú dostačujúce z hľadiska presnosti určenia polohy, cenovo dostupné a väčšina z nich obsahuje aj možnosť navigácie, ktorá je v dnešnej dobe veľmi žiadaná.

Medzi výhody GPS patria[7]:

- Medzi meranými bodmi nemusí byť priama viditeľnosť
- Vysoká presnosť
- Jednotný svetový súradnicový systém
- Poskytovanie trojrozmerných súradníc
- Možnosť získavania polohy kedykoľvek a kdeklvek

Ako každý iný systém, má aj nevýhody:

- Nemožnosť merania v podzemí

- Horšie výsledky pri meraní v husto zastavaných, zalesnených územiach a úskych dolinách
- Potrebná priama viditeľnosť na družice

4.1 Princíp získavania polohy

Presnú polohu na Zemi môžeme získať metódami **trilaterácie**[20]. Podľa toho, či sa snažíme určiť aj vertikálnu pozíciu, použijeme buď **2D** alebo **3D** verziu tejto metódy. K tomu aby bolo možné uskutočniť tieto výpočty, je potrebné:

- Poznať pozíciu najmenej troch satelitov pre 2D a štyroch pre 3D trilateráciu
- Vzdialenosť medzi prijímačom a každou družicou

GPS prijímač získava tieto informácie analýzou rádiových signálov vysiIelaných zo satelitov. V presne určenom čase satelity odosielajú digitálny signál (pseunáhodný kód). Prijímač tento signál začína analyzovať v tom istom okamžiku. Rozdiel medzi odoslaním a prijatím signálu je čas cesty. Vynásobením tejto hodnoty rýchlosťou svetla dokážeme určiť dĺžku trajektórie.

Synchronizácia času jednotlivých satelitov a prijímačov je uskutočnená pomocou atómových hodín v nainštalovaných v každom satelite. Prijímač disponuje kremíkovými hodinami ktoré neustále resetuje. Skúma príchodzie signály zo štyroch a viac satelitov a meria svoju nepresnosť. Presne nastavená hodnota času spôsobí, že sa všetky prijímané signály vyrovnajú v jednom bode. Táto hodnota je hodnotou, udržovanou atómovými hodinami všetkých družíc. Prijímač nastaví svoju hodnotu času na danú hodnotu a je čerstvo zosynchronizovaný s atómovými hodinami všetkých družíc.

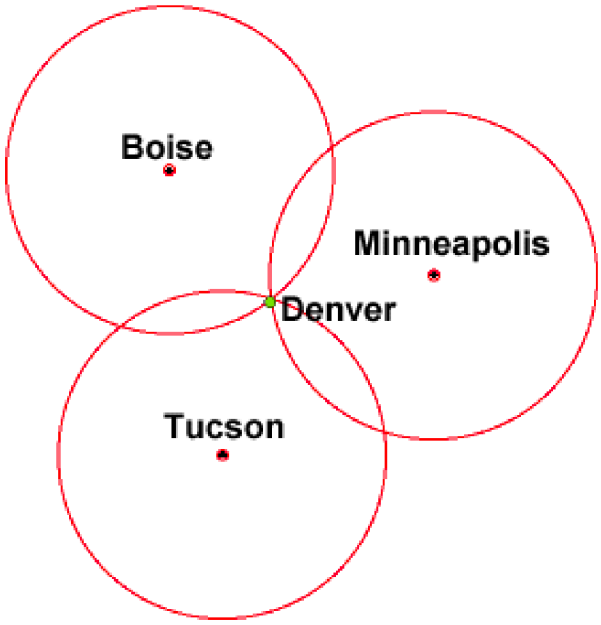
4.1.1 2D Trilaterácia

Pre určovanie polohy na Zemi bez potreby získať aj vertikálnu polohu sa používa metóda 2D trilaterácie. Vysvetlenie tejto metódy uskutočníme pre lepšie pochopenie na príklade.

Predstavme si, že sa nachádzame na neznámom mieste na Zemi. Máme ale k dispozícii zariadenie, ktoré nám dokáže získať vzdialenosť od konkrétneho miesta. Zariadenie nám oznámi, že sa nachádzame X kilometrov od miesta A. Táto informácia je pre nás ale nedostačujúca, nakoľko teraz vieme že sa môžme nachádzať na ktoromkoľvek mieste vzdialenom X kilometrov od miesta A. Množina možných polôh tvorí kružnicu o polomere X kilometrov so stredom v bode A. Po získaní vzdialenosti Y od druhého miesta B vytvoríme ďalšiu kružnicu okolo miesta B s polomerom Y kilometrov. Tým sa nám počet možných polôh obmedzil na dve, ktorými sú priesečníky týchto dvoch kružníc. Pre finálne zistenie polohy určíme pomocou zariadenia vzdialenosť Z aj od tretieho bodu C, ktorá nám rovnakým spôsobom vykreslí ďalšiu kružnicu. Priesečník týchto troch kružníc nám určí jednoznačnú polohu zariadenia.

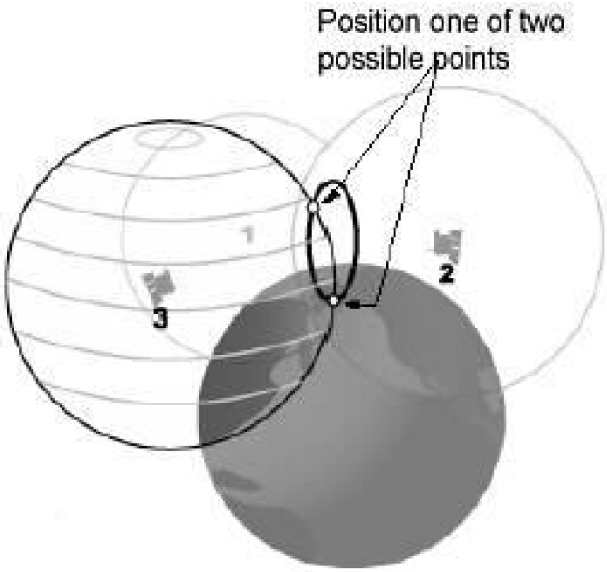
4.1.2 3D Trilaterácia

Narozdiel od 2D trilaterácie, pomocou jej trojrozmernej varianty dokážeme zisťovať aj vertikálnu polohu. Nepracuje s kružnicami, ale s plochami gulí.



Obrázek 4.1: Ukážka princípu 2D trilaterácie

Princíp je veľmi podobný jej 2D variante s tým rozdielom, že si nepredstavujeme kružnice ale gule a neurčujeme vzdialenosť od miest na Zemi ale vzdialenosť od družíc. Predstavme si, že sme vzdialení X kilometrov od satelitu A. Množina možných polôh tvorí povrch gule o polomere X so stredom v bode A. Ak poznáme vzdialenosť Y od satelitu B, vzniknú nám dve gule, ktorých priesečník tvorí kružnicu k. Kružnica k nám tvorí novú množinu potenciónálnych polôh. Pridaním tretej gule so vzdialenosťou (polomerom) Z a stredom v družici C získame množinu iba dvoch bodov, ktoré sú presečníkom kružnice k a novovzniknutej gule so stredom v bode C. Jeden z týchto dvoch bodov leží na Zemi, druhý je v kozme. Za štvrtú gulu môžme považovať samotnú Zem (nakoľko v kozme sa nachádzať nemôžeme), čím sme jednoznačne určili horizontálnu, ale aj vertikálnu polohu. Pre lepšiu predstavivosť je uvedený aj demonštračný nákres na obrázku, ktorý znázorňuje 3 imaginárne gule (A, B, C) a štvrtú guľu ako Zem.



Obrázek 4.2: Ukážka princípu 3D trilaterácie

4.2 Štruktúra systému

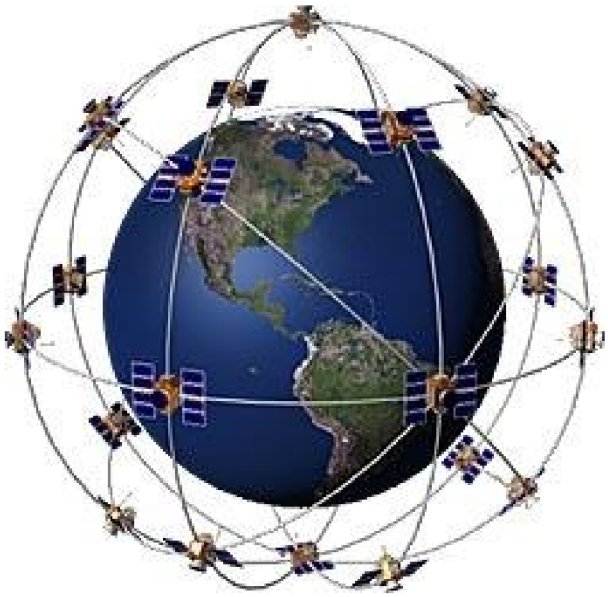
Štruktúru systému GPS tvoria tri základné časti, ktoré sú previazané iba presným časom. Sú nimi **kozmický**, **riadiaci** a **užívateľský** segment[34]. Ich bližšiemu popisu sa budeme venovať v ďalších podkapitolách.

4.2.1 Kozmický segment

Segment je tvorený sústavou družíc rozmiestnených na obežnej dráhe. Pozostáva z 24 družíc, z ktorých je 21 navigačných a tri plnia úlohu záložných satelitov. Každá družica obieha Zem približne raz za 12 hodín a majú voči nej stálu pozíciu. Samotná konštelácia (rozmiestnenie) je tvorené šiestimi obežnými dráhami a štyrmi družicami na každej z nich, ktoré majú sklon 55 stupňov vzhľadom k polárnej rovine. Toto usporiadanie poskytuje užívateľom signál zo štyroch až dvanástich družíc na ktoromkoľvek mieste na Zemi. Každá družica je vybavená štyrmi atómovými hodinami, akumulátormi, batériami a solárnymi panelmi. Približná váha takejto družice je 900 kilogramov.

4.2.2 Riadiaci a kontrolný segment

Riadiaci segment je zodpovedný za riadenie celého globálneho polohového systému. Jeho hlavnou úlohou je aktualizovať údaje obsiahnuté v družicových navigačných správach. Tento segment je tvorený systémom štyroch monitorovacích staníc rozmiestnených okolo celého sveta. V Colorade (USA) je umiestnená hlavná riadiaca stanica. Okrem týchto staníc sys-



Obrázek 4.3: Trajektórie družíc obiehajúcich okolo Zeme

tém obsahuje aj tri stanice umožňujúce vysielať na družice údaje o ich obežných dráhach a nastavení hodín. Takéto informácie družice môžu prijímať i niekoľkokrát denne. Monitorovacie stanice merajú signály vysielané družicami a získané údaje prenášajú do Hlavnej riadiacej stanice. Tu sú na základe prijatých výsledkov vypočítané presné údaje obežných dráh (**efemeridy**) a korekcie hodín pre jednotlivé družice, ktoré sú prenesené na vysielacie stanice. Vysielacie stanice potom minimálne jedenkrát denne vysielajú aktualizované efemeridy s údajmi o nastavení hodín na jednotlivé družice. Tieto družice potom prostredníctvom rádiových signálov vysielajú obežné dráhy a presný čas do GPS príjímačov.

4.2.3 Užívateľský segment

Užívateľský segment sa skladá z GPS prijímačov, užívateľov, vyhodnocovacích nástrojov a postupov. Na základe prijatých signálov z družíc uskutočňujú GPS prijímače predbežné výpočty polohy, rýchlosti a času. Pre výpočet všetkých štyroch súradníc (tri priestorové a čas) je potrebné naviazať signál s minimálne štyrmi družicami. Takéto prijímače sú používané pre navigáciu, stanovovanie polohy, presného času a podobne.

Kapitola 5

Analýza požiadaviek

Cieľom našej práce je vytvoriť systém pre monitorovanie pohybu vozidiel na základe GPS. Takýto systém je využiteľný najmä pre menšie spoločnosti zaoberajúce sa rozvozom produktov, transportom osôb a podobne. Je vhodná najmä na kontrolu príchodov, odchodov, meškaní, priebežných meškaní a kontrolu absolvovanej vzdialenosti.

Pre aplikáciu tohto typu je nevyhnutné zariadenie s GPS a mobilným pripojením, ktorých je na súčasnom trhu veľké množstvo. Naše zariadenie musí taktiež obsahovať možnosť zobrazenia používateľského rozhrania, s ktorým by bolo možné interaktívne pracovať. Týmto sa nám počet potencionálnych zariadení výrazne zmenšil. Po krátkej úvahe sme zistili, že zariadení ponúkajúcich tieto možnosti je v súčasnosti veľmi bežné a vlastní ho takmer každý človek. Týmto zariadením je inteligentný mobilný telefón ktorý ponúka spojenie prijateľného užívateľského rozhrania, možnosť nadviazania internetového spojenia, či už pomocou mobilného dátového spojenia alebo za pomoci bezdrôtového pripojenia, a taktiež dostatočne presnú možnosť zisťovania aktuálnej polohy za pomoci GPS. Toto zariadenie je maximálne blízke a dostupné užívateľovi, čo vytvára vzhľadom na aplikáciu výbornú podporu rozšíriteľnosti.

V prípade, že by sme mali vyriešenú otázku výberu vhodného zariadenia získavajúceho GPS dáta, mali by sme zvážiť akým spôsobom by mal pracovať systém, ktorý tieto dáta bude spracovávať. V prípade, že by sme uvažovali nad deskopovou aplikáciou nainštalovanou na počítačoch osôb, ktoré by mali prístup k zobrazeniu spracovaných dát, bolo by veľmi obtiažne vytvoriť takú verziu systému, ktorá by dokázala pracovať na akejkoľvek verzii ktorejkoľvek platformy. Museli by sme taktiež uvažovať o inštalácii samotnej aplikácie na každý počítač, pripojenie aplikácie na server, ktorý by spájal mobilnú a deskopovú aplikáciu a taktiež by nám v prípade chýb aplikácie technická podpora nedokázala v krátkom čase pomôcť a vyriešiť vniknutý problém. Z týchto dôvodov sa ako lepšie riešenie ukazoval webový informačný systém bežiaci na doméne užívateľa. Takýto systém bude prístupný zovšadiaľ a nie je potreba brať ohľad na rozdielne operačné systémy užívateľov.

Z vyššie uvedeného vyplýva, že najlepším spôsobom ako navrhnúť systém je, že sa bude skladať z dvoch častí, a to mobilnej a webovej. Primárnym účelom mobilnej aplikácie bude poskytovanie GPS koordinátov aktuálnej polohy zariadenia a ich odosielanie na server, ktorý ich bude spracovávať a ukladať. Zobrazovanie týchto dát v rôznych formách bude úlohou webovej aplikácie. V tejto obasti existuje veľa možností, či už real-time zobrazovanie na mape alebo textová reprezentácia údajov. V otázke odosielania GPS koordinátov mobilnou aplikáciou na server sa nám objavuje závažný problém, a to možné výpadky navigačného alebo internetového spojenia. Z hľadiska straty GPS signálu bohužiaľ nie je možné plnohodnotne reagovať a nahradiť chýbajúce koordináty. Je tak možné iba zvoliť čo

najideálnejšie zotavenie po chybe a to formou ignorovania absencie tohto signálu a dotazovaním sa ďalších dát až pokým nedôjde k opätovnému naviazaniu spojenia. V prípade výpadku internetového spojenia je situácia takmer plnohodnotne riešiteľná. Tým, že používame zariadenie disponujúce internou pamäťou, je možné neúspešne odoslané koordináty uchovávať. Po opätovnom naviazaní internetového signálu je možné tieto dáta v správnom poradí odoslať na server na spracovanie. V tomto prípade nedošlo k žiadnej strate dát, ale iba k zdržaniu ich vyhodnotenia. Jedinou nevýhodou tohto prístupu je, že stav údajov v databázy nebude vypovedať realite v každom okamihu, avšak zaoberáme sa riešením chybového stavu a nie správnou funkčnosťou, takže pre nás táto situácia je prijateľná.

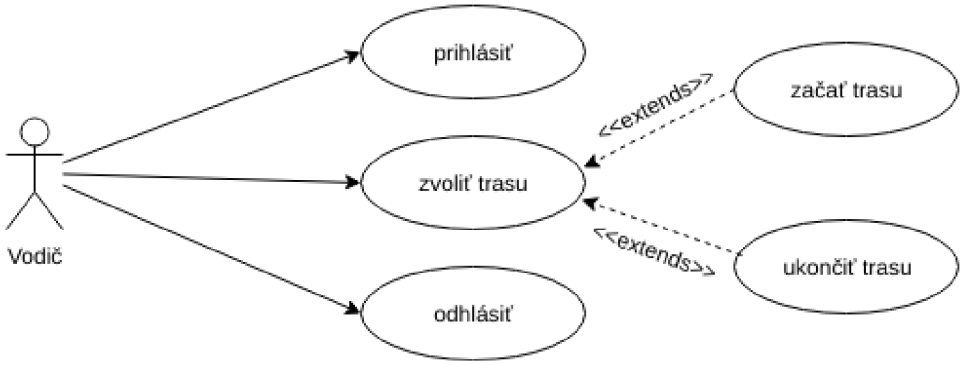
Primárnym cieľom aplikácie je monitorovanie pohybu vozidiel na vopred naplánovaných trasách. V tomto by som chcel ozrejmiť, že webová časť aplikácie slúži nielen k zobrazeniu dát o danej trase, ale aj k jej plánovaniu, uchovávaniu, správe užívateľov, vodičov a cieľových či štartových adries. K tomuto systému by mali mať prístup iba užívatelia určený pre správu trás a vyššie postavení zamestnanci spoločnosti využívajúcej daný softvér. Iná situácia je ale u mobilnej aplikácie, ktorá je primárne určená pre šoférov a samotných aktérov trasy. Mal by byť do nej zabezpečený prístup a mala by ponúkať všetky trasy, ktoré sú naplánované pre daného vodiča. Jeho úlohou by malo byť iba jednoduché zvolenie trasy zo zoznamu, jej začatie a ukončenie. Iná funkcionalita v tomto prípade potrebná nie je. Bolo by taktiež ideálne aby bolo možné inteligentný telefón plnohodnotne používať aj počas chodu aplikácie. Tá by tak mala bežať na pozadí zariadenia a podávať iba chybové hlášky o strate či už vyššie spomenutého GPS alebo internetového signálu. Takúto vlastnosť dokážeme docieliť spustením určitých častí aplikácie na pozadí a následnou kontrolou aktivity, prípadne reštartom tejto časti ak by náhodou došlo k násilnému uvoľneniu pamäti operačným systémom inteligentného telefónu.

Na obrázkoch je možné vidieť *use-case* diagramy mobilnej aj webovej aplikácie. Mobilná aplikácia plní úlohu získavania dát, teda aj jej funkcionalita nie je rozsiahla. Nakoľko užívatelia mobilnej aplikácie (vodiči) môžu byť aj ľudia s minimálnymi skúsenosťami s technológiami, je potrebné aby bolo ovládanie jednoduché, jasné a nevyžadovalo od užívateľa príliš informácii. Po prihlásení by mala aplikácia zobrazovať prehľad neukončených a nezačatých trás ktoré sú naplánované pre prihláseného vodiča. Intiutívnym výberom trasy sa vodičovi zobrazia všetky podrobnosti o trase a možnosť spustiť alebo zastaviť trasu podľa aktuálneho stavu.

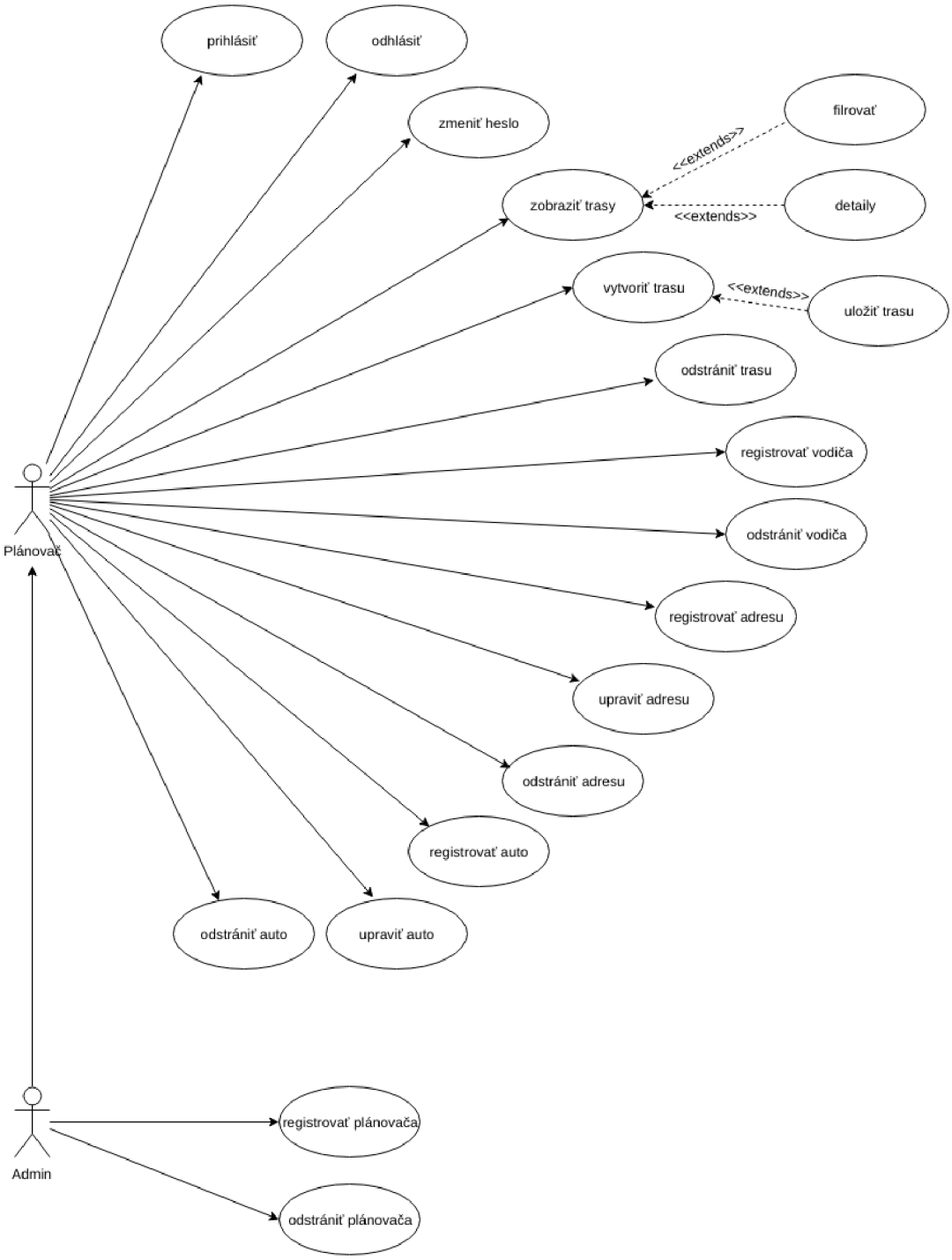
Ako už bolo vyššie spomenuté, aplikácia prevádza užívateľa jednotlivými krokmi plánovania. Každá obrazovka reprezentuje jeden ucelený celok, ktorým môže byť napríklad zadávanie štartu, cieľa, prejazdového bodu alebo obrazovky s prehľadom už naplánovaných informácii o trase s tlačidlom pre uloženie a potvrdenie zmien.

V prvom kroku je užívateľ vyzvaný pre výber vodiča a auta, ktoré sú možné zadať formou výberu zo zoznamu. Taktiež je potrebné zadať počet prejazdných bodov. Ďalej plánovač vyberá štartovú a cieľovú adresu zo zoznamu existujúcich adries, v prípade že neexistuje, zaregistruje ju a opakuje krok. K jednotlivým adresám je potrebné v správom poradí vyplniť aj plánovaný čas štartu a konca trasy. Po potvrdení zadaných údajov je užívateľ presmerovaný, podľa počtu naplánovaných prejazdných bodov na zadávanie prvého prejazdného bodu kde plánovač zadá adresu prejazdu zo zoznamu, čas príchodu a odchodu. V prípade, že je potrebných viac prejazdov sa tento krok opakuje viackrát podľa počtu zadaných prejazdných bodov na úvodnej obrazovke. Na posledný krok, na ktorom je prehľad naplánovanej trasy čakajúcej na potvrdenie, je plánovač presmerovaný v prípade nulového počtu prejazdov hneď po úvodnom kroku alebo po zadaní informácii o poslednom prejazde. Táto stránka obsahuje informácie o vodičovi, aute, štarte, cieli, počte prejazdov a v prípade

existencie prejazdov aj informácie o nich. Logika naplánovaných časov trasy musí byť logicky správne usporiadaná, inak je užívateľ upozornený. Rovnako je kontrolované, aby sa ľudskou chybou nenaplánovali dve rôzne trasy pre jedného vodiča alebo auto. Pred potvrdením je možné danú trasu uložiť aj ako predpripravenú pre ďalšie použitie, kde bude uchovávané poradie a počet adries, vodič, ŠPZ auta a počet prejazdov. Užívateľ pri neskoršom využití tejto trasy doplní iba časové údaje každého bodu a uloží trasu. V prípade potreby menších zmien je možné každý údaj upraviť.



Obrázek 5.1: Use-case diagram mobilnej aplikácie



Obrázek 5.2: Use-case diagram webovej aplikácie

Kapitola 6

Návrh

V tejto kapitole si uvedieme detailný návrh celého systému a voľbu implementačných prostriedkov. Budeme sa zaoberať jak na prvý pohľad jasnými časťami, tak aj časťami, ktoré boli pri analýze požiadavkov označené ako problematické.

6.1 Návrh systému

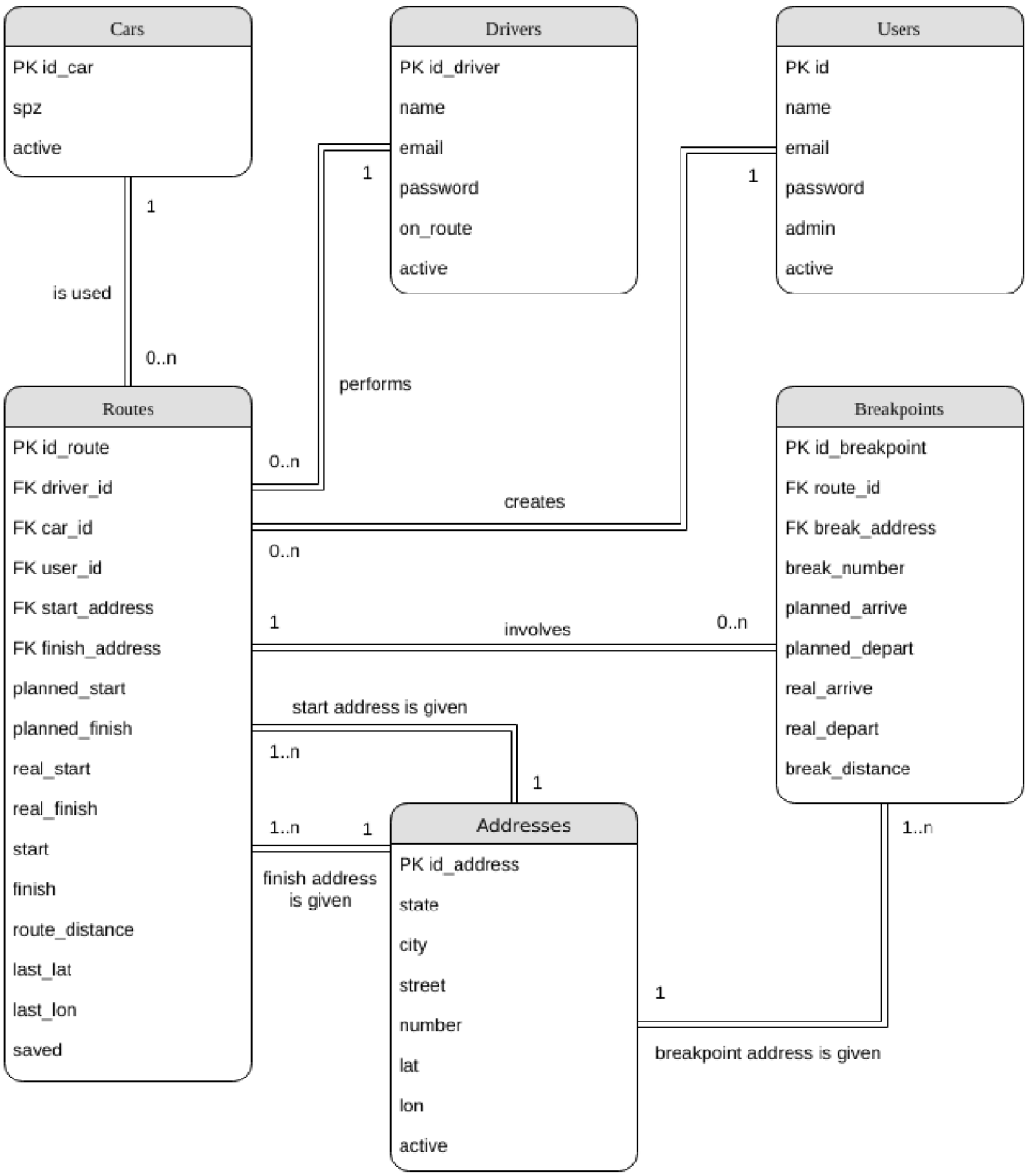
Základná štruktúra a funkčnosť systému bola popísaná v kapitole analýzy požiadaviek. Zhodli sme sa, že systém sa bude skladať z dvoch častí, ktoré budú pripojené pomocou logiky a databázy bežiacej na serveri. Bolo však potrebné zamyslieť sa nad detailnou skladbou databázy a taktiež doplniť bežnú funkcionalitu nevyhnutnú pre intuitívne a prirodzené zaobchádzanie so systémom. Pre uchovávanie dát bola zvolená databáza o počte šiestich tabuliek. Využitá bola relačná databáza podporujúca prácu na serveri MySQL.

Nakoľko naša aplikácia pracuje s veľkým množstvom dátumov, časov a adries, bolo by ideálne disponovať istou znovupoužiteľnosťou týchto dát vo forme cudzích klúčov v jednotlivých tabuľkách. Otázka adries bola elegantne vyriešená zapracovaním tabuľky obsahujúcej všetky minimálne jedenkrát využité adresy do systému. Takto uložené dáta jednotlivých adries je možné znovuvyužiť pre plánovanie ďalších trás, čo sa v reálnych situáciach objavuje pomerne často. Takto uložené adresy je možné pomocou cudzieho kľúča priradiť akémukoľvek množstvu trás.

Otázka týkajúca sa znovupoužiteľnosti časov a dátumov jednotlivých trás sa ukázala ako nereálne riešenie. Každá trasa je z pohľadu časov jedinečná, aj keď z hľadiska adries môže ísť o identické trasy. Z tohto dôvodu je potrebné vytvoriť pre každú trasu nový záznam v tabuľke trás s informáciami o štartovej aj cieľovej adrese, ich plánovanými časmi a dátumami a rovnako dátami zobrazujúcimi reálne časy a vzdialenosti danej trasy.

Jednou z hlavných informácii ktoré sa nám podarilo v rámci skúmania potrieb užívateľa získať je schopnosť aplikácie registrovať aj prejazdové miesta. Tieto miesta môžu slúžiť ako kontrolné body alebo aj ako zastávky pri prípadnom rozvoze tovaru alebo osôb. Bolo by totiž veľmi obmedzujúce pre užívateľa ak by mohol naplánovať iba štart a cieľ ak pritom v reálnych situáciách je bežné, že má vozidlo určité zastávky. Pre poskytnutie tejto vlastnosti bolo potrebné pridať tabuľku obsahujúcej prejazdné body trasy. Rovnako je potrebné určiť aj adresu prejazdu, ktorá je v tabuľke prejazdných bodov uložená rovnakým princípom ako to funguje u štartovej a cieľovej adresy. Všetky vyššie popísané informácie sú graficky znázornené pomocou ER diagramu, v ktorom sú viditeľné aj tabuľky obsahujúce informácie

o vozidlách, užívateľoch a vodičoch. Každý užívateľ (plánovač, administrátor) môže byť zároveň aj vodičom.



Obrázek 6.1: ER diagram webovej aplikácie

Ako je možné si odvodiť, tabuľka obsahujúca základné informácie o trasách má názov **Routes**, tabuľka obsahujúca dáta adries **Addresses**. **Breakpoints** je tabuľka obsahujúca prejazdné body jednotlivých trás a tabuľky **Cars**, **Drivers** a **Users** obsahujú dáta používaných áut, registrovaných vodičov a užívateľov (plánovačov, administrátorov).

Zaujímavé sú taktiež položky *last_lat* a *last_lon* v tabuľke routes, ktoré reprezentujú koordináty posledného miesta, po ktoré bola meraná vzdialenosť. Slúžia teda k výpočtu prejdenej vzdialenosti každej trasy.

6.2 Návrh webovej aplikácie

Pri návrhu webovej časti systému sme považovali za podstatné, aby daná aplikácia neslúžila iba pre zobrazenie, uloženie a vytvorenie jednotlivých trás. Cieľom bolo dosiahnuť aby slúžila užívateľom ako ucelený systém s plnou funkcionalitou. V prvom rade bolo potrebné zabezpečenie prihlasovacieho systém z dôvodu rôznych útokov na sieti, čo bol aj jeden z dôvodov rozhodnutia sa využiť framework pre implementáciu systému. Po dôkladnej analýze bol zvolený PHP framework *Laravel* ponúkajúci okrem iných funkcií práve túto možnosť. Tento systém podporuje registráciu užívateľov, ale aj obnovenie hesiel s možnosťou odosielania notifikačných emailov.

Aplikácia okrem činností spojených s trasami disponuje aj ďalšími možnosťami, ako je prehľad a správa užívateľov, ktorú má k dispozícii iba administrátor. Ďalej rovnako prehľad a správa registrovaných adries, áut a vodičov. V prípade adries a áut má administrátor aj plánovač možnosť upraviť jednotlivé položky.

Pre návrh užívateľského rozhrania sú využité klasické webové technológie ako *HTML*, *CSS* a *JavaScript*. Je taktiež využitá grafická šablóna **Now UI Kit**, ktorá spadá pod CSS framework **Bootstrap 4**, ktorý sa v súčastnosti veľmi využíva pre tvorbu Front-endu a uľahčuje webovým dizajnérom pracné štýlovanie HTML elementov. Pomocou tejto šablóny je vytvorený celkový vzhľad aplikácie. Pre jej využitie sme sa rozhodli najmä z dôvodu výbornej podpory vo forme dokumentácie.

6.3 Návrh mobilnej aplikácie

Návrh mobilnej aplikácie je do značnej miery ovplyvnený malým množstvom funkcií, ktoré aplikácia poskytuje užívateľovi. Mobilná aplikácia je pre nás iba prostriedkom pre získavanie aktuálnej pozície zariadenia. Ako implementačný prostriedok bol zvolený jazyk *Java* a pre popis užívateľského rozhrania bol zvolený značkovací jazyk *XML*. Mobilná aplikácia je vyvíjaná pre platformu **Android** najmä kôli jeho jednoznačnému postaveniu lídra vo svete operačných systémov pre inteligentné telefóny. Aj z tohto dôvodu je aplikácia vyvíjaná v prostredí **Android Studio SDK**, ktoré disponuje všetkými potrebnými knižnicami.

V prípade optimálnych podmienok, kedy by nenastávali žiadne výpadky GPS alebo internetového signálu, by aplikácia obsahovala triedy pre reprezentáciu jednotlivých obrazoviek, triedu pre získavanie GPS koordinátov a metódy pre odosielanie dát na server. Riešenie týchto problémov je načrtnuté už v analýze požiadaviek. Počas štúdia operačného systému Android bolo preukázané uvoľňovanie zdrojov s nižšou prioritou v prípade zaplnenia operačnej pamäte, prípadne aktivácia automatického šetrenia batérie, čo má za následok vypínanie aplikácie bez zásahu užívateľa. Tento problém je možné vyriešiť iteratívnou kontrolou činnosti aplikácie a spustením kľúčových aktivít aplikácie (získavanie a odosielanie GPS koordinátov na server) na pozadí tak, aby to užívateľa v bežnej práci s inteligentým telefónom neobmedzovalo.

6.4 Návrh serverovej logiky

Návrhom serverovej logiky je myslený mechanizmus spracovávania prijatých dát získaných z mobilnej aplikácie. Tvorí ju sada skriptov v jazyku PHP, ktoré prijímajú požiadavky typu *POST* a následne v spolupráci s databázou vyhodnocujú prijaté dáta a ukladajú potrebné

informácie. Pre implementáciu tejto sady budú využité prostriedky frameworku Laravel a to možnosť smerovania požiadaviek na jednotlivé *Kontroléry*, ktoré predstavujú dané skripty.

Je potrebné si určiť aké typy požiadavkov bude potrebné spracovávať a aké dáta budú obsahovať. Ako prvé sa budeme zaoberať odosielaním bežných požiadaviek kedy nedochádza k chybám či už absenciou GPS alebo internetového spojenia. Takáto požiadavka by mala obsahovať identifikátor trasy na ktorú sa dáta viažu a aktuálnu polohu vo forme súradníc. Tieto údaje sú porovnávané s dátami uloženými v databáze. V prípade, že sú splnené určité podmienky vyplývajúce z aktuálnej situácie danej trasy, sú dáta (časy a vzdialenosť) uložené do prislúchajúcich položiek záznamov. Pri úspechu či neúspechu je odoslaná odpoveď mobilnej aplikácii. Druhou variantou požiadavky je ukončenie trasy zo strany mobilnej aplikácie, kedy sú odoslané dáta rovnakého formátu na server, ktorý ich spracováva rovnakým spôsobom ako pri bežnom spracovaní dát. Rozdiel je, že odpoveď aplikácie obsahuje informáciu o ukončení monitorovania trasy a zastaví sa iteratívne získavanie koodrinátov a ich odosielanie na server.

V prípade výpadku internetového spojenia na strane mobilnej aplikácie, sú neúspešne odoslané dáta po znovunaviazaní pripojenia odoslané na server ako požiadavka obsahujúca počet neúspešne odoslaných koordinátov, identifikátor trasy a samotné koordináty. Logika na serveri takúto požiadavku spracováva ako niekoľko bežných požiadaviek po sebe, avšak na základe počtu vynechaných požiadaviek dokáže vyhodnotiť ich časové zdržanie a v prípade potreby zadať správny čas (napríklad príchodu do cieľa) do položiek záznamov databázy. Výpadok GPS signálu nie je možné stranou serverovej logiky riešiť, nakoľko takáto požiadavka nie je ani na server odosielaná.

Kapitola 7

Implementácia

V nasledujúcej kapitole sa budeme zaoberať podrobnou implementáciou jednotlivých častí systému. Cieľom bola prehľadnosť a efektívnosť implementácie aby sme sa vyhli zbytočne opakujúcemu sa kódu. Pri implementácii sme využívali najmä vedomosti objektovo-orientovaného programovania.

7.1 Implementácia webovej časti aplikácie

Ako už bolo spomenuté, pre implementáciu webovej časti aplikácie bol použitý PHP framework Laravel. Pre implementáciu užívateľského rozhrania bol použitý CSS framework Bootstrap 4 a jeho šablóna Now UI Kit, ktorý ponúka nielen príjemné užívateľské rozhranie ale aj responzivitu celej aplikácie.

7.1.1 Prihlásenie a registrácia užívateľov a vodičov

Pre vytvorenie prihlasovacieho a registračného systému sme využili nástroje frameworku Laravel, ktorý ponúka automatické vytvorenie relačných tabuliek databázy uchovávajúcich informácie o užívateľoch (tabuľka *users*) ale aj obnovovaniach hesiel (tabuľka *password_resets* - táto tabuľka nie je zobrazená v ER diagrame aplikácie, nakoľko funkcionalitou stojí mimo ostatných tabuliek). Tento nástroj pre nás automaticky vytvorí registračný a prihlasovací formulár a v prípade zabudnutého hesla ponúka možnosť obnovenia pomocou notifikačného emailu.

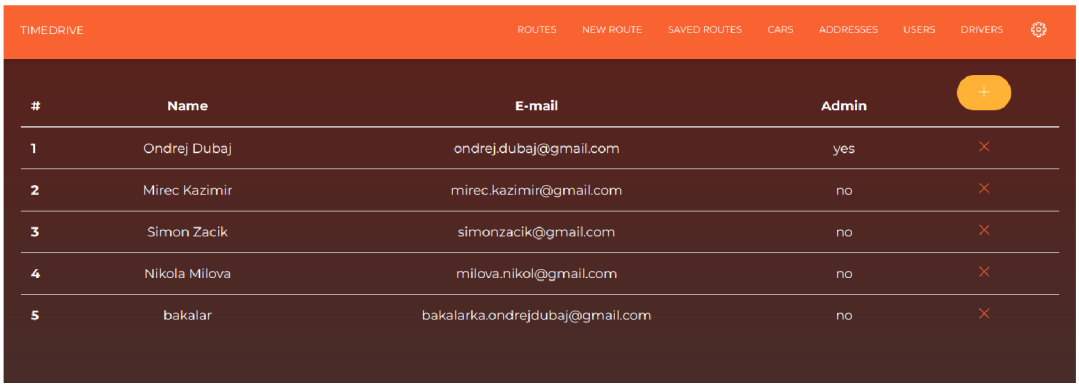
V našej aplikácii bolo potrebné pôvodné vlastnosti tohto systému pozmeniť, aby odpovedali naším potrebám. Prihlasovací systém z hľadiska funkčnosti ostal nezmenený, kde je neautorizovaný užívateľ po navštívení danej domény automatický presmerovaný na prihlasovací formulár ponúkajúci možnosť obnovenia hesla v prípade zabudnutia. Prihlásenie do systému prebieha pomocou emailu a hesla, ktoré je v databáze zašifrované pomocou východiskovej Laravel funkcie pre uchovávanie hesiel *bcrypt()*. V prípade zhody mena a hesla s niektorým záznamom v tabuľke users je užívateľ presmerovaný na domovskú stránku systému obsahujúcu prehľad trás.

Pri využití funkcie "zabudnutého hesla" je užívateľ vyzvaný k zadaniu emailu, na ktorý mu bude odoslaná správa obsahujúca odkaz pre obnovenie hesla. Zadaný email musí existovať v tabuľke *users*, a taktiež sa vytvorí záznam v tabuľke *password_resets*, ktorý obsahuje čas vytvorenia záznamu, email užívateľa a jedinečný hash vytvorený funkciou *bcrypt()*. Odkaz je zabezpečený tak, že ako svoj parameter obsahuje práve tento zašifrovaný reťazec, ktorý sa jednoznačne viaže k emailu a jeho doba platnosti je 60 minút. Výsledkom je, že

takýto odkaz môže slúžiť iba jedinému užívateľovi a nikto iný ho nemôže použiť. Po presmerovaní na danú URL je užívateľ vyzvaný k zadaniu svojho emailu a hesla, ktoré je potrebné ešte raz potvrdiť. Po správnom vyplnení formuláru je užívateľ automaticky prihlásený a presmerovaný na domovskú stránku.

Vlastnosti registračného systému sú výrazne pozmenené oproti štandardnému nastaveniu frameworku. Bežne je možná registrácia ktoréhokoľvek návštevníka aplikácie alebo stránky, avšak náš systém má slúžiť iba privilegovaným užívateľom, preto bolo potrebné toto nastavenie zmeniť v samotnom jadre Laravelu. Registrácia aj odregistrácia užívateľov (plánovačov) a vodičov je možná iba autorizovanému užívateľovi, pričom registrácia a odregistrácia plánovača je privilégium iba užívateľa s označením administrátor. Tieto dve registrácie sú od seba oddelené aby nedochádzalo k omylom. Pri oboch registráciach je nutné zadať iba meno a email užívateľa/vodiča. Po odoslaní formuláru je vytvorený záznam v príslušnej tabuľke (plánovač v tabuľke *users*, vodič v tabuľke *drivers*), kde nie je vyplnené pole hesla. Taktiež je odoslaná správa na daný email obsahujúca rovnako zabezpečený odkaz ako v prípade obnovenia hesla. Ďalej princíp nastavenia hesla funguje rovnako, avšak rozdiel medzi registráciou plánovača a vodiča je v tom, že plánovač je prihlásený a presmerovaný na domovskú stránku, ale vodičovi sa zobrazí iba stránka potvrdzujúca nastavenie hesla. Je to z dôvodu, že vodič nie je oprávnený pristupovať do systému, ale iba do mobilnej aplikácia, kde dokáže ovplyňovať iba záznamy určené jeho profilu.

Pre prehľad registrovaných vodičov a plánovačov sú vytvorené tabuľky s možnosťou odregistrovať danú osobu. Časť zobrazujúca plánovačov je prístupná iba užívateľom s právmi administrátora.



Obrázek 7.1: Ukážka tabuľky užívateľov

7.1.2 Prehľad trás

Prehľad trás nám umožňuje zobraziť všetky naplánované prebiehajúce alebo ukončené trasy a základné informácie o nich. Týmito informáciami sú myslené meno vodiča, ŠPZ auta, adresa štartu, adresa cieľa, plánovaný čas štartu, plánovaný čas ukončenia, stav jazdy a aktuálne zdržanie vozidla. Všetky tieto informácie sú získané z požiadaviek na tabuľky databázy pomocou príkazu *SELECT* a syntaktickej konštrukcie jazyka SQL slúžiacej k spojovaniu výsledkov *JOIN*. Zdržanie vozidla je určené zdržaním na poslednom meranom mieste (štarte, prejazdovom bode, cieli) alebo v prípade, že zdržanie na akutálne najbližšom bode presiahne zdržanie na poslednom meranom mieste (rozdiel aktuálneho času a plánovaného

start	finish	stav
0	0	Nezačatá
1	0	Na ceste
1	1	Ukončená

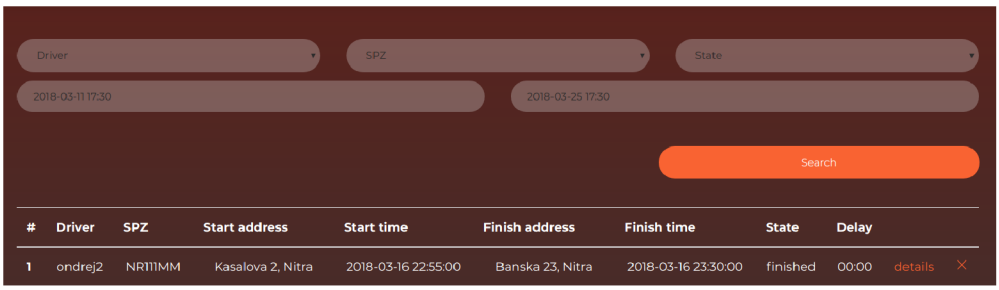
Tabulka 7.1: Tabuľka popisujúca stav danej trasy vzhľadom na stav premenných **start** a **finish**

príjazdu je väčšie ako zdržanie na poslednom bode), je toto zdržanie považované za meškanie vozidla. Stav danej trasy je určený z dvoch položiek každého záznamu v tabuľke *routes*. Tými položkami sú **start** a **finish**, ktoré môžu nadobúdať hodnoty *true* a *false*. Pre lepšiu interpretáciu kombinácií jednotlivých hodnôt položiek a stavov uvádzam prehľadnú tabuľku.

Stav *"nezačatá"* reprezentuje, že vozidlo zatiaľ neopustilo miesto štartu, alebo že trasa ešte vôbec nebola spustená. Stav *"na ceste"* reprezentuje, že vozidlo opustilo miesto štartu a ešte nedorazilo do miesta cieľu. *"Ukončená"* logicky reprezentuje poslednú možnosť kedy auto s vodičom úspešne dorazili do cieľa a ukončili trasu.

Systém taktiež umožňuje filtrovanie jednotlivých záznamov trás pre lepšiu prehľadnosť a možnosť rýchlejšieho vyhľadávania dát. Je implementované ako formulár, ktorého dáta sú odosielané *POST* požiadavkou a využité pre upravenie SQL požiadavky získavajúceho požadované zánamy z databázy. Filtrovanie dát je možné uskutočniť na základe nasledujúcich položiek:

- Mena vodiča
- ŠPZ auta
- Stavu záznamu
- Dátum plánovaného štartu trasy
- Dátum plánovaného ukončenia trasy



Obrázek 7.2: Ukážka filtru a časti tabuľky záznamov

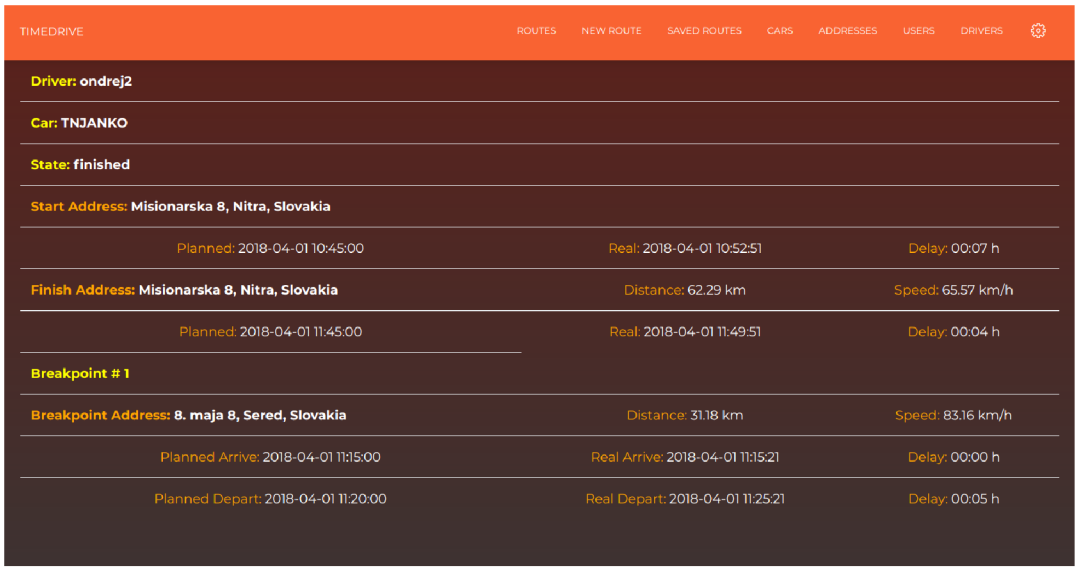
V prehľade trás sa avšak zobrazujú údaje iba dvoch týždňov záznamov, a to tých, ktoré majú svoj štart naplánovaný v posledných siedmych dňoch a záznamov, ktoré majú svoje plánované ukončenie v najbližších siedmych dňoch. Záznamy sú takto prehľadnejšie a užívateľ sa zbytočne nemusí zdržovať príliš starými alebo aj príliš dopredu naplánovanými

záznamami. V prípade potreby zobraziť dáta z iného časového intervalu, je možné využiť vyššie spomenutú možnosť filtrovania dát.

V tejto časti aplikácie je taktiež možné v prípade chyby jednoducho vymazať určité trasy aj s ich naplánovanými prejazdmi jednoduchým kliknutím na tlačidlo. Je tiež možné zobraziť detaily každej trasy, ktoré popíšem v nasledujúcej kapitole.

7.1.3 Detaily trasy

V detailoch trasy je možné vidieť všetky položky databázy reprezentujúce danú trasu. Sú tu informácie o plánovaných aj reálnych časoch na všetkých kontrolných bodoch, ich zdržaniach, adresách, vzdialenostiach od štartu a tiež priemernú rýchlosť dosiahnutú na absolvovanej trase. Rovnako je tu možné vidieť informácie o vodičovi, aute a stave. Informácie sú získané SQL požiadavkou, kde sú z rozdielu plánovaných a reálnych časov vypočítavané jednotlivé zdržania. Meranie vzdialenosti sa uskutočňuje najmä pomocou koordinátov získaných z mobilnej časti aplikácie a položiek *last_lat* a *last_lon* v tabuľke *routes*.



Obrázek 7.3: Ukážka detailov trasy

7.1.4 Vytvorenie novej trasy

Proces vytvorenia novej trasy je priamo závislý na počte prejazdných bodov danej trasy. Tento proces je prakticky vypĺňaním formuláru, v ktorom je väčšina elementov implementovaná ako možnosť výberu zo zoznamu, prípadne je k dispozícii možnosť zobrazenia kalendára s časovými hodinami pre presné zvolenie daného času a dátumu.

Implementačne je takýto formulár vytvorený z viacerých menších, ktoré na seba logicky nadväzujú. Takúto implementáciu sme zvolili najmä z dôvodu väčšej prehľadnosti a tak aj menšej chybovosti užívateľov. Z našich štúdií sme zistili, že ak má užívateľ pred sebou dvadsať rôznych položiek na vyplnenie, je pre neho menej prirodzené sa v tom zorientovať ako keď sú tieto elementy usporiadané v piatich krokoch, kde každý krok predstavuje ucelenú

časť trasy. Pre kontrolu a prehľadnosť sú pred uložením trasy zobrazené všetky informácie, ktoré užívateľ zadal a má tak možnosť ich kontroly, prípadne opravy.

V prvom kroku formuláru je užívateľ vyzvaný k zadaniu vodiča, auta a počtu plánovaných prejazdných bodov. Výber vodiča a auta je implementovaný pomocou HTML elementu *<select>*, kde možnosti predstavujú zoznam aktívnych áut a vodičov získaných z tabuliek databázy. Textové pole pre počet prejazdov umožňuje užívateľovi zadať iba číselné hodnoty, teda je implementovaný ako *<input type="number">*. V tomto kroku je okrem vyššie zmienených údajov potrebné zadať adresu štartu s jeho plánovaným dátumom a časom a tiež adresu cieľu s časovými údajmi. Výber adresy je v oboch prípadoch taktiež implementovaný výberom z možností, rovnako ako je to u výberu vodiča či auta. Pre implementáciu pomocného kalendára pre jednoduché určenie dátumu a času bola využitá Javascript funkcia *datetimepicker()*. Táto funkcia je naviazaná na štandardný textový HTML element vstupu pomocou identifikátora. Pri kliknutí na dané pole sa zobrazí dialógové okno, v ktorom je po krokoch možné určiť dátum a čas.

```
return Validator::make($data, [
    'driver' => 'required|string|not_in:0',
    'spz' => 'required|string|not_in:0',
    'start_address' => 'required|string|not_in:0',
    'finish_address' => 'required|string|not_in:0',
    'breakpoints_number' => 'required|numeric|min:0',
    'finish_time'=> 'required|date_format:Y-m-d H:i|after_or_equal:start_time',
]);
```

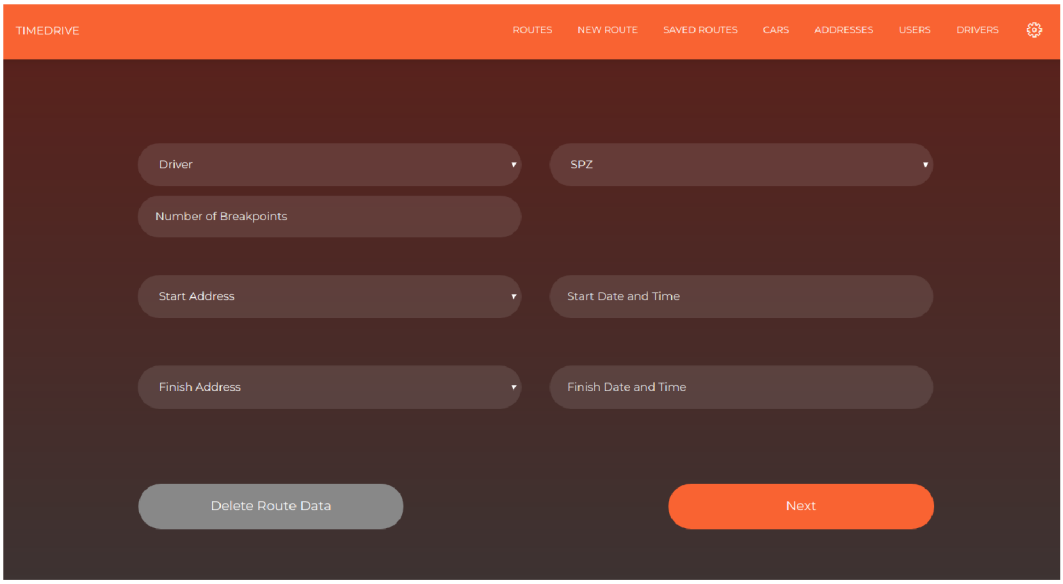
Obrázek 7.4: Ukážka kódu validátora

Po vyplnení údajov a pokračovaní na ďalší krok prebieha najskôr validácia obsahu tohto formuláru, kontroluje sa správne zvolenie vodičov, áut, adries, počet prejazdov, ktorý nesmie nadobúdať záporné hodnoty, logická časová následnosť zadaných dátumov a časov a taktiež prebieha kontrola, či dané auto a vodič nemajú v časovom rozmedzí záznamu naplánované iné trasy. Validácia formuláru prebieha za pomoci objektu typu **Validator**, ktorý je súčasťou Laravelu a poskytuje možnosti rýchlej kontroly údajov a prípadný návrat so zobrazením príslušnej chybovej hlášky. V prípade úspešnej validácie, je užívateľ presmerovaný na ďalší krok formuláru a vyplnené údaje sú uložené pomocou vstavanej funkcie **session()**, ktorá predstavuje superglobálnu premennú *$_SESSION*.

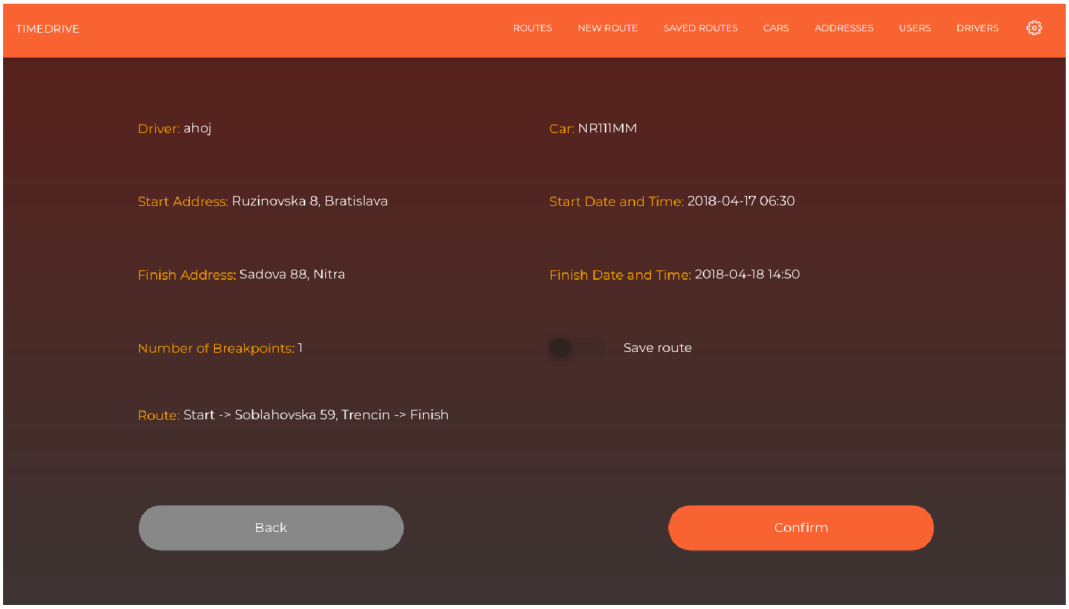
Po úspešnom dokončení úvodného kroku nás môže aplikácia presmerovať na dva rôzne kroky podľa počtu prejazdných bodov. V prípade zadania nuly, uskutoční sa presmerovanie na posledný krok na ktorom je vidieť prehľad vyplnených údajov. V prípade potreby upraviť tieto údaje, je možné využiť možnosť vrátiť sa na k predchádzajúcim krokom a upraviť potrebné polia. Ak údaje sedia, sú po potvrdení vytvorené záznamy v príslušných tabuľkách databázy.

Ak užívateľ zadal minimálne jeden prejazdný bod, je presmerovaný na stránku s ďalším formulárom, kde je potreba vyplniť adresu bodu, predpokladaný dátum a čas príchodu na danú adresu a predpokladaný dátum a čas odchodu. Vyplňovanie údajov prebieha rovnako ako na úvodnom formulári. Taktiež je pre validáciu použitý objekt typu **Validator** a jednotlivé časy sú kontrolované vzhľadom na všetky už zadané časy. Ukladanie jednotlivých krokov formuláru je riešené taktiež pomocou funkcie **session()**. Takýto formulár sa užíva-

teľovi zobrazí toľkokrát, koľko zvolil na začiatku prejazdných bodov. Po úspešnom vyplnení posledného je užívateľ presmerovaný na posledný krok, ktorý je kontrolný a bol popísaný vyššie.



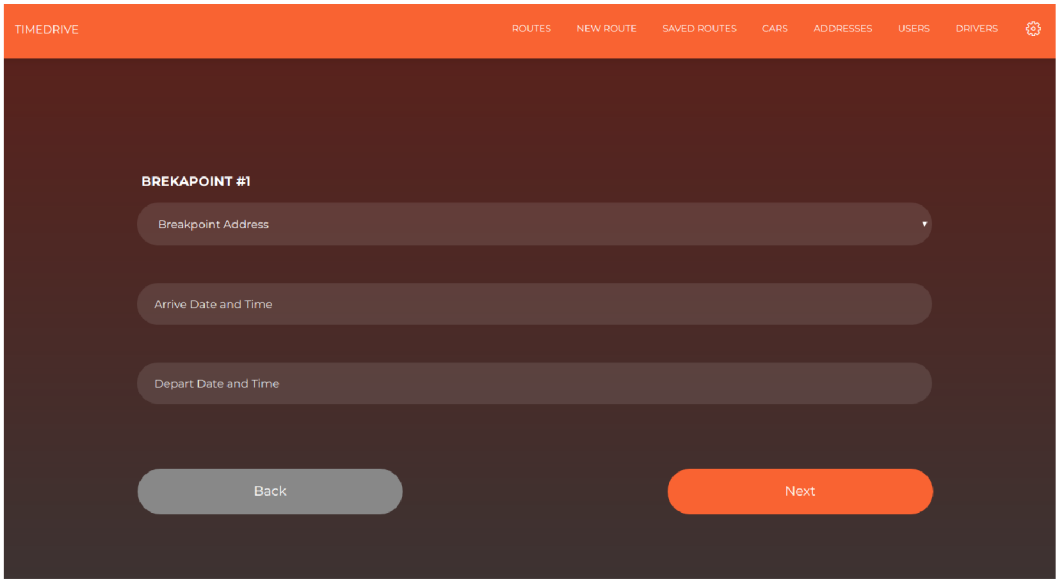
Obrázek 7.5: Ukážka prvého kroku vytvárania trasy



Obrázek 7.6: Ukážka posledného kroku vytvárania trasy

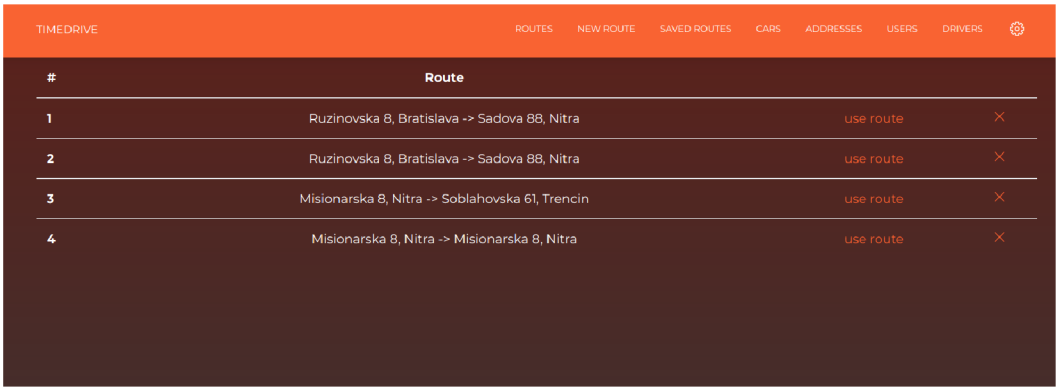
Možnosť vrátiť sa o krok späť je implementovaná taktiež pomocou funkcie **session()**, kedy si do pola zaznamenávame históriu navštívených URL. Samotné ukladanie a čítanie

jednotlivých URL adries funguje princípom zásobníka, kedy v prípade návratu prechádzame na URL uloženú na vrchole, ktorú odstraňujeme. V prípade posúvania sa dopredu, vždy ukladáme URL, z ktorej sa presmerovávame ďalej, na vrchol. Týmto princípom sa dokážeme vyhnúť chybám, ktoré sa stávali pri viacnásobnom použití Laravel funkcie *back()*, ktorá zaznamenáva iba posledne navštívenú URL. Ide teda o vlastnú implementáciu tejto funkcie s odstránením chýb a optimalizáciou pre "refresh" stránky, kedy je vstavaná funkcia rovnako nepoužiteľná, nakoľko pri tejto situácii je posledne navštívená URL totožná súčasnej.



Obrázek 7.7: Ukážka zadávania prejazdového bodu

Užívateľ má v prípade potreby možnosť uložiť si danú trasu medzi obľúbené. Táto možnosť je implementovaná z dôvodu možného pravidelného opakovania tých istých trás v iných časoch. Pri uložení tejto trasy je do položky **saved** v tabuľke *routes* uložená hodnota true. Takto označené záznamy je možné použiť pre rýchlejšie naplánovanie rovnakej trasy, akú sme kedysi už absolvovali. Pri samotnom využití sa nám automaticky vyplnia všetky údaje formuláru okrem plánovaných časov. Každú jednu položku je možné upraviť v prípade, že danú trasu absolvuje iný vodič, prípadne iné vozidlo. Predvyplnenie údajov sa týka taktiež adries štartu, cieľa a všetkých prejazdných bodov v danom poradí, v akom boli usporiadané v predchádzajúcej trase.



Obrázek 7.8: Ukážka uložených trás

7.2 Implementácia mobilnej časti aplikácie

Implementácia mobilnej časti aplikácie sa skladá z tried jazyka *Java*, ktoré zabezpečujú funkčnosť aplikácie, a súborov v jazyku *XML* reprezentujúcich užívateľské rozhranie. Aplikácia je sprístupnená iba užívateľom, ktorí sú registrovaní ako vodiči. Ponúka nám tri typy obrazoviek a je možné ju používať iba v prítomnosti internetového spojenia, nakoľko každá požiadavka je spracovávaná na serveri. Jednotlivé obrazovky nám reprezentujú jeden druh aktivity, kde pri prepnutí na inú aktivitu sa pôvodná ukončuje a začína sa nasledujúca. Toto nám zabezpečuje knižný objekt typu *Intent*.

Obrázek 7.9: Obrazovka s prehľadom naplánovaných trás

Prvá aktivita reprezentuje prihlasovací formulár, kde vodič zadá svoj email a heslo, ktoré si nastavil pomocou webovej časti aplikácie. Po zadaní údajov sú tieto dáta pomocou knižnice **com.android.volley** odoslané požiadavkou typu *POST* na server, kde prebehne kontrola správnosti a je odoslaná naspäť odpoveď vo formáte *json*. V prípade neúspechu je na obrazovku vypísaná chybová hláška (*"nesprávne meno"*/*"nesprávne heslo"*). V prípade úspechu sú spolu s odpoveďou odoslané aj dáta reprezentujúce naplánované a nedokončené trasy daného vodiča. Po obdržaní odpovede je začatá aktivita zobrazujúca tieto dáta. Každá

trasa je prerezentovaná plánovaným dátumom a časom začiatku a konca trasy a takisto ich adresami. Táto aktivita slúži pre výber trasy, ktorú chceme začať alebo ukončiť.

Po vybraní trasy sa následne odosiela *POST* požiadavka na server, ktorá obsahuje identifikátor danej trasy a server nám v odpovedi vráti detailnejšie údaje o trase, ktoré zobrazí v novej aktivite, aby užívateľ mohol skontrolovať správnosť vybranej trasy. V prípade voľby nesprávnej, je možné vrátiť na predchádzajúcu aktivitu pomocou tlačidla *"Naspäť"*. Po overení správnosti trasy je možné trasu začať alebo ukončiť v závislosti na jej stave. Tým sa povinnosti zo strany vodiča voči aplikácii končia, nakoľko o samotné odosielanie koordinátov, komunikáciu so serverom a kontrolu aktivity sa už stará systém. Taktiež aplikácia neumožňuje vodičovi spustiť dve trasy súčastne, čo by mohlo viesť k dátovým kolíziam. Túto vlastnosť kontrolujeme pomocou položky *on_route* v tabuľke *drivers*. Nakoľko je aplikácia závislá na internetovom pripojení, je potrebné týmto spojením disponovať pri začatí trasy a rovnako aj pri jeho ukončení. Mimo týchto podmienok dokáže aplikácia fungovať bez internetového pripojenia, i keď táto možnosť je pôvodne implementovaná z dôvodu možnej straty spojenia spôsobenou okolitými podmienkami, dokáže plnohodnotne nahradiť bežnú prevádzku.

Obrázek 7.10: Obrazovka s detailnými informáciami o vybranej trase

7.2.1 Automatické odosielanie koordinátov

Získavanie samotných koordinátov zariadenia zabezpečuje trieda *GPSTracker.java*, ktorá využíva funkcie objektu **LocationManager**, ktorej funkcie pre získanie aktuálnej pozície sú volané v pravidelných časových intervaloch. Pre implementáciu iteratívneho správania

je použitý objekt typu **Handler**. V jednom intervale, ktorý je v našej aplikácii nastavený na hodnotu 30 sekúnd, dochádza najskôr k získaniu koordinátov a následne k ich odoslaniu štandardným spôsobom pomocou knižnice *com.android.volley* spolu s indentifikátorom trasy. Nedostupnosť GPS signálu aplikácia rieši chybovým hlásením, avšak interval neustále pokračuje až pokiaľ nie je užívateľom ukončená trasa. V prípade chyby pri odosielaní koordinátov, ktorá je vo väčšine prípadov spôsobená absenciou internetového pripojenia, sú tieto dáta ukladané do internej pamäte pomocou rozhrania **SharedPreferences**, ktorá tieto dáta dokáže perzistentne uložiť i v prípade násilného ukončenia aplikácie. Postup je taký, že pri dosiahnutí intervalu sa skontroluje stav internetového pripojenia. V prípade, že internetové spojenie aj naďalej absentuje, uložia sa aj novo získané súradnice do internej pamäte. V prípade nadviazania spojenia, sú najskôr odoslané koordináty uložené v internej pamäti spolu s identifikátorom danej trasy a počtom koordinátov, ktoré boli v predošlých iteráciach neúspešne odoslané. Tento počet slúži k určeniu časového zdržania dát v prípade, že by v priebehu výpadku signálu došlo k dosiahnutiu určitého kontrolného bodu a aby jednotlivé časy neboli touto chybou ovplyvnené. Po úspešnom odoslaní dát je interná pamäť uvoľnená a aplikácia pokračuje v štandardnom režime. Pri ukončení trasy sa rovnako kontroluje, či boli všetky koordináty úspešne odoslané (či je pamäť prázdna) a až následne sa trasa ukončí odoslaním posledných koordinátov na server a ukončením pravidelných iterácii. Je to z dôvodu, že užívateľ môže dosiahnuť cieľ v krátkom čase a hneď aplikáciu ukončiť. Toto v prípade dlhého intervalu medzi odoslaniami môže spôsobiť chybné informácie v databáze z dôvodu, že posledné koordináty sa pred ukončením nestihli odoslať.

Celý tento vyššie popísaný postup je implementovaný ako trieda, ktorá rozširuje triedu **Service**. Tá nepredstavuje štandardnú aktivitu alebo triedu, ale činnosti bežiace na pozadí operačného systému. Táto implementácia je zvolená z dôvodu, že môže dôjsť k násilnému ukončeniu aplikácie a z toho dôvodu aj ukončenia odosielania samotných koordinátov na server. Pre takéto prípady má aplikácia implementovaný alarm pomocou triedy ktorá rozširuje triedu **BroadcastReceiver**. Táto trieda nám ponúka možnosť kontrolovať aktivitu procesu bežiaceho na pozadí v nepravidelných intervaloch a v prípade jeho neaktivity ho znova naštartovať. Tento alarm je aktivovaný spolu so spustením trasy a je ukončený až s jej ukončením. Z toho vyplýva, že samotné odosielanie koordinátov nie je možné dlhodobejšie ukončiť alebo prerušiť iným spôsobom ako riadnym vypnutím, čo je v našom prípade veľmi žiadaná vlastnosť. Nakoľko kontrolovanie aktivity bežiacej na pozadí prebieha v nepravidelných intervaloch so stredom 30 sekúnd, je možné, že po násilnom ukončení dójde k väčšiemu časovému intervalu medzi odosielaniami, avšak odchýlka sa pohybuje rádovo v sekundách, čo je v prípade presnosti údajov zanedbateľné.

7.3 Implementácia serverovej logiky

Funkciou serverovej logiky je spracovávanie požiadaviek prichádzajúcich z mobilnej aplikácie. Medzi tieto požiadavky môžme radiť aj overenie prihlasovacích údajov a následné odoslanie dát obsahujúcich zoznam trás daného vodiča, prípadne odoslanie detailov už vybratej trasy. Skripty obsluhujúce tieto požiadavky komunikujú s databázou a preposielajú odpovede mobilnej aplikácii. V týchto prípadoch sa jedná zväčša o krátke a jednoduché PHP skripty implementované ako *Kontroléry* webovej časti aplikácie. Dôležitou vlastnosťou je, že serverová logika odosiela odpoveď vždy, i v prípade chybného požiadavku. Je to z dôvodu, že aplikácia očakáva nejakú odpoveď a na základe nej reaguje.

7.3.1 Spracovanie bežných požiadaviek

Zaujímavejšou a taktiež náročnejšou časťou logiky je spracovávanie požiadaviek obsahujúcich koordináty. Pri obdržaní prvého požiadavku obsahujúceho koordináty a identifikátor trasy je v tabuľke *drivers* priradená hodnota identifikátora trasy do položky *on_route*, čo reprezentuje, že vodič používa aplikáciu a začal určitú trasu. Na základe prijatého identifikátora sú z databázy získané informácie o stave trasy (položky *start*, *finish*). Tu sa nám naskytujú tri možnosti:

- **Trasa nezačala** - táto informácia nám oznamuje že vozidlo ešte neopustilo okruh miesta štartu a tým pádom nebola zaznamenaná ani žiadna vzdialenosť. V tomto prípade sa získané súradnice použijú pre výpočet vzdialenosti medzi miestom štartu a aktuálnym miestom. V prípade, že je táto geometrická vzdialenosť väčšia ako 100 metrov, uloží sa aktuálny dátum a čas do príslušnej položky záznamu databázy a položka *start* sa nastaví na hodnotu *true*. V tomto prípade sa taktiež daná vypočítaná vzdialenosť pripočíta k položke *distance*, ktorá má na začiatku hodnotu *nula*. Taktiež sa získané koordináty uložia do položiek *last_lat* a *last_lon*, ktoré reprezentujú poslednú pozíciu, po ktorú bola meraná vzdialenosť. V opačnom prípade, kedy rozdiel medzi súradnicami štartu a aktuálnymi súradnicami je menší ako 100 metrov, nevykonáva sa žiadna z vyššie popísaných činností. V oboch prípadoch je mobilnej aplikácii odoslaná odpoveď vo formáte *json*.

- **Trasa beží** - touto informáciou sa rozumie, že vozidlo opustilo okruh štartu a ešte nedorazilo do miesta cieľa. Vozidlo sa preto pre nás nachádza na bližšie nešpecifikovanom mieste. Pomocou databázového požiadavku vieme presne určiť medzi ktorými kontrolnými bodmi sa vozidlo nachádza, ktorý už opustilo alebo na ktorom momentálne je. Na základe tohto si od databázy vyžiadame prvý prejazdný bod (tabuľka *breakpoints*), ktorý nemá niektorú z časových položiek (*real_arrive*, *real_depart*) zadanú. V prípade, že takýto bod nájdeme, pracujeme s ním podľa toho, či nie je vyplnená časová položka príchodu (vozidlo ešte nedorazilo na kontrolný bod, čaká sa na jeho príjazd) alebo odchodu (vozidlo sa momentálne nachádza na mieste kontrolného bodu, čaká sa na jeho odjazd). Na základe tejto informácie sú nastavené aj jednotlivé podmienky pre kontrolu vzdialenosti súradníc kontrolného bodu a aktuálne získaných súradníc (pred príchodom sa očakáva, kedy bude vzdialenosť medzi aktuálnou polohou a kontrolným bodom menšia ako 100 metrov, pred odchodom kedy bude väčšia). V prípade platnosti danej podmienky sa zaznamená časové razítko, prebieha pripočítanie vzdialenosti (v prípade príchodu aj uloženie tejto vzdialenosti do položky *break_distance*) a odosiela sa odpoveď. Ak nám databázový požiadavok vráti prázdny výsledok (trasa nemá prejazdné body alebo sa vozidlo nachádza na poslednom úseku trasy, teda pred cieľom), nasledujúce porovnávania sa uskutočňujú so súradnicami cieľa. V prípade splnenia podmienky (vzdialenosť koordinátov aktuálnej polohy a cieľa je menšia ako 100 metrov) nastáva zaznamenanie časového razítka, prípočet vzdialenosti, nastavenie položky *finish* na hodnotu *true* a odoslanie odpovedi. Týmto sa daná trasa považuje za ukončenú.

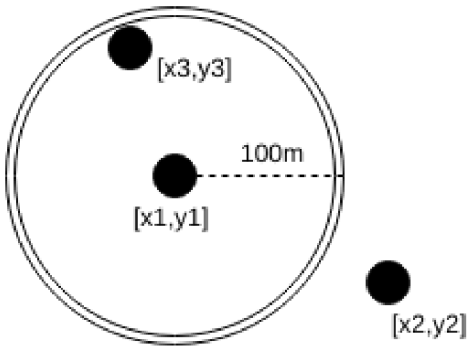
- **Trasa je ukončená** - túto informáciu získame na základe položiek *start* a *finish*, kedy sú obe nastavené na hodnoty *true*. V tomto prípade serverová logika nevykonáva žiadne ďalšie požiadavky na databázu a odosiela odpoveď mobilnej aplikácii.

7.3.2 Ukončenie trasy

Požiadavka na ukončenie trasy obsahuje rovnaké typy dát ako je to pri bežnej požiadavke. Rozdiel je iba v cieľovej URL, ktorá predstavuje iný skript. Tento skript avšak začína nastavením hodnoty *on_route* daného vodiča na hodnotu *nula*. Tým sa pre vodiča pomyselne ukončuje daná trasa. Následne je zavolaný skript pre spracovanie bežného požiadavku, nakoľko aj táto ukončujúca požiadavka obsahuje koordináty pre kontrolu. Celý proces kontroly je teda identický s bežným procesom, ktorý je popísaný vyššie.

7.3.3 Znovuodoslanie po chybe internetového pripojenia

Požiadavka určená pre nápravu chybového stavu, ktorý bol spôsobený výpadkom internetového pripojenia či inej chybe, obsahuje identifikátor trasy, počet neúspešne odoslaných súradníc a samotný zoznam súradníc. Postup spracovania týchto súradníc je totožný so spracovaním bežnej požiadavky avšak zoznam prijatých súradníc sa spracováva iteratívne pre každú dvojicu koordinátov. Zakaždým sa kontrolujú všetky tri možnosti v ktorých sa môže daná trasa vyskytovať a súradnice sú spracovávané v poradí v akom boli zaznamenané pomocou mobilnej aplikácie. Ako v štandardnom prípade logika reaguje na daný stav v ktorom sa trasa nachádza. Jediný rozdiel je v zaznamenávaní časovej značky, od ktorej je v prípade potreby odpočítaný čas zdržania danej dvojice súradníc spôsobený výpadkom spojenia. Výpočet tohto zdržania prebieha na základe rozdielu počtu neúspešne odoslaných súradníc a hodnoty indexu danej dvojice súradníc. K tejto hodnote je pripočítaná hodnota jedna a výsledok je vynásobený dĺžkou intervalu medzi dvoma odosielaniami súradníc, čo v našom prípade je 30 sekúnd. Rovnako ako v prípade bežnej požiadavky je po uskutočnení všetkých potrebných akcií odoslaná odpoveď mobilnej aplikácii.



Obrázek 7.11: Ukážka okruhu zaznamenávajúceho GPS koordináty

7.3.4 Výpočet vzdialenosti

Tento výpočet sa uskutočňuje pomocou vzorca, ktorý počíta geometrickú vzdialenosť dvoch bodov na povrchu Zeme určenými súradnicovým systémom.

$$a = sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(lon1 - lon2)$$

$$distance = acos(a) * 60 * 1.1515 * 1.609344$$

Premenné *lat1*, *lon1*, *lat2*, *lon2* predstavujú geografickú šírku a dĺžku dvoch rozdielnych bodov a sú udávané v radiánoch. Vzorec pre výpočet bol prebraný z literatúry uvedenej v zdrojoch. Premenná *distance* je výsledná hodnota vzdialenosti udávaná v kilometroch.

Kapitola 8

Testovanie

Pre testovanie aplikácie neboli využité žiadne automatické testy, čoho dôvodom bola vysoká obtiažnosť vytvárania týchto testov. Samotný systém bol testovaný po častiach počas jeho vývoja, čím boli chyby jednotlivých častí zachytené už v ich počiatku a neboli zanesené do ďalšej funkcionality.

8.1 Testovanie webovej časti aplikácie

Toto testovanie spočiatku prebiehalo zväčša samotným vývojárom, nakoľko boli testované jednotlivé moduly funkcionality. Avšak po úspešnom dokončení webovej časti bolo testovanie uskutočňované najmä bežnými užívateľmi, ktorí sa nepodieľali na vývoji. Práve po konzultáciách s užívateľmi boli pridané určité prvky do aplikácie, ako napríklad filter pre zobrazovanie jednotlivých trás, či možnosť uloženia danej trasy pre ďalšie použitie, možnosti upravenia údajov adresy alebo auta. Najhlavnejšou požiadavkou bolo však viackrokové zadávanie trasy. Pôvodne bolo totiž zadávanie trasy na jednej obrazovke, čo spôsobovalo omyly na strane užívateľa a taktiež to z grafického hľadiska, pri udávaní väčšieho počtu prejazdných bodov, nevyzeralo prívetivo. Výhodou však bol prehľad všetkých dát na jednom mieste. Na ich podnety bolo zadávanie trasy a ostatné výhrady prepracované a pri ďalšom testovaní vládla spokojnosť. Samotnú funkčnosť a chybové stavy testoval vývojár webových aplikácií, ktorý sa na vývoji tohto systému nepodieľal. Tento tester mal taktiež prístup do samotnej databázy, takže mohol prispôsobovať jednotlivé záznamy svojim potrebám. Boli odhalené mierne odchýlky vo výpočtoch zdržania vozidiel, avšak tieto chyby boli rýchlo odstránené.

8.2 Testovanie mobilnej časti aplikácie

Pri testovaní mobilnej aplikácie sme sa zamerali najmä na funkčnosť na rôznych verziách operačného systému Android. Testovanie prebiehalo na nasledujúcich zariadeniach:

- Lenovo K6 (Android 6.0.1)
- Asus Zenfone 3 (Android 7.0)
- Samsung Galaxy S4 mini (Android 4.4)
- Samsung Galaxy S5 mini (Android 5.1)

- Samsung Galaxy A3 (Android 7.0)

Testovanie na zariadeniach prebehlo úspešne s výnimkou zariadenia s verziou operačného systému Android 4.4. Je to pravdepodobne z dôvodu zastaralosti tejto verzie a podľa prieskumov, zastúpenie tejto verzie vo svete sa pohybuje pod hranicou desať percent a neustále klesá. Preto sme sa viac nefunkčnosťou aplikácie na tejto verzii operačného systému nezapodievali.

Taktiež boli na mobilnej aplikácii testované základné úkony ako prihlásenie a odhlásenie užívateľa, vybratie, začatie a ukončenie trasy. Testovaná bola možnosť spustiť 2 trasy v jeden okamih a rovnako bolo skúmané správanie aplikácie pri výpadku jedného z potrebných signálov pre prácu aplikácie.

Náhodným testovaním sa prišlo na ukončovanie aplikácie v prípade dochádzania zdrojov. Tento problém sa však nevyskytoval na zariadeniach s operačným systémom verzie 7.0, teda je predpoklad, že to má jadro operačného systému tento problém v najnovších verziách už vyriešený. Avšak pre staršie verzie bol z tohto dôvodu a z dôvodu možného násilného ukončenia aplikácie užívateľom implementovaný alarm popísaný v časti implementácie.

8.3 Testovanie celého systému

Testovanie celého systému prebiehalo v teréne, kde boli vývojárom pozorované logovacie hlášky získavané z mobilného zariadenia a taktiež reakcie serverovej logiky na prichádzajúce požiadavky. Boli odhalené mierne nezrovnalosti v určovaní príchodu na určité miesto, avšak bolo to spôsobené príliš veľkou vyžadovanou presnosťou koordinátov na strane serveru a neschopnosťou mobilného zariadenia poskytnúť túto presnosť. Taktiež boli odhalené logické chyby pri určovaní prejdenej vzdialenosti, avšak po krátkom opravení a doladení jednotlivých častí logiky aplikácia fungovala bezchybne vrátane zotavenia sa po chybe. Jediná záležitosť, ktorá môže spôsobovať problémy je už spomínaná neschopnosť zariadenia poskytnúť dostatočnú presnosť GPS spojenia, prípadne dlhodobý výpadok tohto signálu, ktorý môže spôsobiť aj nekonzistentnosť dát v samotnej databáze. Toto je už ale z pohľadu systému neovplyvniteľné a jediné riešenie je zaobstaranie si presnejšieho a kvalitnejšieho zariadenia pre získavanie koordinátov aktuálnej pozície.

Kapitola 9

Možnosti rozšírenia

Daný systém ponúka viacero možností rozšírenia a pokračovania v prípade väčšieho množstva času na ich realizáciu. Jednalo by sa najmä o isté rozšírenia funkčnosti systému, nie zmeny nastávajúceho.

Prvou možnosťou rozšírenia je zobrazenie jednotlivých vozidiel na mape pre real-time sledovanie pozície. Pre toto rozšírenie je možné využiť *Google Maps* v kombinácii s technológiou *Ajax*, kde za pomoci rôznych knižníc a funkcií, ktoré poskytuje jazyk JavaScript by sa na mape vykreslili pozície všetkých áut, ktoré sú momentálne na ceste. Každé auto by mohlo byť označené svojím ŠPZ značením a po prejdení myši na dané auto by sa zobrazili aj ďalšie informácie o trase. Jedinou nevýhodou tohto rozšírenia je dlhodobejší výpadok internetového spojenia, kde by aplikácia zobrazovala už neaktuálnu pozíciu vozidla. V tomto prípade by bolo na mieste riešiť globálne pokrytie internetového signálu, čo však presahuje hranice tohto systému aj práce.

Ďalšou možnosťou rozšírenia je využitie iného zariadenia pre zber a odosielanie GPS dát na server. Súčasný trh ponúka veľa možností vo forme športových hodiniek, rôznych GPS prijímačov, navigácií a ďalších zariadení. Ideálne by bolo mať k dispozícii zariadenie disponujúce malou obrazovkou a tlačidlami pre výber trasy, ktoré by obsahovalo slot pre kartu SIM poskytujúcu mobilné dátové pripojenie. Nakoľko by sa jednalo o zariadenie určené pre takéto využitie, disponovali by sme aj podstatne vyššou presnosťou GPS signálu. Ideálnym zariadením by boli GPS navigácie ktorejkoľvek značky, ktoré by okrem zaznamenávania danej trasy mohli poskytovať aj navigáciu vozidla a plánovanie a zaznamenávanie trás. Reálnejšie je však pre túto potrebu využiť zariadenia v súčasnosti najčastejšie využívané pre turistiku či *GeoCaching*.

Obrázek 9.1: Zariadenie určené k *GeoCaching*u

Kapitola 10

Záver

Cieľom našej práce bol vývoj systému pre monitorovanie pohybu vozidiel. Daný systém by mal byť využiteľný v reálnej praxi a mal by slúžiť prevažne menším spoločnostiam, ktoré nedisponujú, či už z finančných alebo iných dôvodov, rozsiahlym systémom pre sledovanie časových údajov jednotlivých vozidiel na trasách. Systém by mal slúžiť ako primárny prostriedok pre zaznamenávanie časov príjazdov, odchodov, zdržaní a vzdialenosti, nakoľko väčšina spoločností v súčastnej dobe využíva zastaralý spôsob ručného zapisovania týchto údajov do papierovej formy, čo môže viesť k chybám a taktiež neprehľadnosti, nakoľko týchto tlačív môže byť veľké množstvo. Systém ponúka prehľadný spôsob uloženia a zobrazenia dát v reálnom čase dostupný kdekoľvek, kde sa nachádza internetové pripojenie.

Najväčšia výhoda systému spočíva v jeho jednoduchosti a prehľadnosti. Užívateľ nie je nútený vypisovať zdĺhavé formuláre na každom prejazdnom bode, ale iba pri plánovaní trasy, ktorá môže byť znovuvyužiteľná. Samotnú tvorbu trás a prehľad výsledkov zabezpečuje webová aplikácia, ktorá intuitívnym spôsobom interpretuje dáta uložené v databáze. Spustenie trasy a zber dát zabezpečuje mobilná časť aplikácie, ktorá je vyvinutá pre platformu Android. Ovládanie mobilnej aplikácie spočíva iba vo výbere, začatí a ukončení danej trasy, takže užívateľ ňou je zamestnaný len minimálne.

Testovaním systému sa preukázala schopnosť zaznamenávania a vyhodnocovania údajov o trase. Užívatelia, ktorí sa podieľali na testovaní boli s výsledkami a ovládaním systému spokojní. Najväčšiu výhodu videli vo využití bežného prostriedku, ako je inteligentný telefón, pre daný účel. V prípade, že by sa niekedy v budúcnosti naskytla príležitosť použiť systém v reálnom fungovaní spoločnosti, rád by som túto možnosť využil a v prípade potreby si vypočul ďalšie požiadavky a rozšírenia na systém, ktoré by bolo možné implementovať.

Počas štúdia a implementácie systému som sa dopodrobna zoznámil so zásadami fungovania GPS, čo je z hľadiska všeobecného prehľadu veľmi prospešné. Taktiež ako veľký prínos vidím v práci s frameworkom Laravel, ktorý bol použitý pre implementáciu webovej aplikácie. Naučil ma novým zásadám programovania webových aplikácií, najmä zoznámenie s návrhovým vzorom MVC. Jednou z úplne nových skúsenosti bol vývoj aplikácii pre zariadenia s operačným systémom Android. Vedomosti objektovo-orientovaného programovania v jazyku Java v kombinácií s tvorbou užívateľského prostredia pomocou značkovanieho jazyka XML a zoznámenie so základnými postupmi vývoja Android aplikácií určite využijem v budúcnosti.

Literatura

[1] Allen, G.: *Android 4*. Computer press Apress, 2012, ISBN 978-80-251-3782-6.

[2] Anonym: About BlackBerry. [Online; navštívené 17.04.2018].
URL https://us.blackberry.com/company

[3] Anonym: BlackBerry. [Online; navštívené 15.04.2018].
URL https://cs.wikipedia.org/wiki/BlackBerry

[4] Anonym: Co je responzivní web a proč ho mít? [Online; navštívené 27.03.2018].
URL http://www.easyweb.cz/proc-responzivni-web

[5] Anonym: CSS Syntax and Selectors. [Online; navštívené 24.03.2018].
URL https://www.w3schools.com/css/css_syntax.asp

[6] Anonym: Differences between Client-side and Server-side Scripting. [Online; navštívené 26.03.2018].
URL https://www.sqa.org.uk/e-learning/ClientSide01CD/page_18.htm

[7] Anonym: Družicové polohové systémy. [Online; navštívené 03.04.2018].
URL http://www.sgs.edu.sk/HTML/gps.htm

[8] Anonym: História mobilných telefónov. [Online; navštívené 01.04.2018].
URL http://www.mobilforum.sk/historia-mobilnych-telefonov-t116.html

[9] Anonym: HTML Elements. [Online; navštívené 27.03.2018].
URL https://www.w3schools.com/html/html_elements.asp

[10] Anonym: Introduction. [Online; navštívené 05.04.2018].
URL https://laravel.com/docs/4.2/introduction

[11] Anonym: Introduction to Laravel PHP Framework Features and Version History. [Online; navštívené 18.04.2018].
URL http://stacktips.com/tutorials/laravel/intro-to-laravel-php-framework-and-features

[12] Anonym: IoC Container. [Online; navštívené 05.04.2018].
URL https://laravel.com/docs/4.2/ioc

[13] Anonym: jQuery Introduction. [Online; navštívené 02.04.2018].
URL https://www.w3schools.com/jquery/jquery_intro.asp

[14] Anonym: Laravel. [Online; navštívené 04.04.2018].
URL https://cs.wikipedia.org/wiki/Laravel

[15] Anonym: The Evolution of GPS. [Online; navštívené 07.04.2018].
URL http://illumin.usc.edu/70/the-evolution-of-gps/

[16] Anonym: Typy web stránok. [Online; navštívené 21.03.2018].
URL https:
//blog.subject.sk/tvorba-web-stranok/web-stranky/typy-web-stranok.html

[17] Anonym: Vznik a história web stránok. [Online; navštívené 20.03.2018].
URL https://dizajn-web.webnode.sk/vznik-a-historia-web-stranok/

[18] Anonym: Windows Phone. [Online; navštívené 11.04.2018].
URL https://cs.wikipedia.org/wiki/Windows_Phone

[19] Anonym: WWW (World Wide Web). [Online; navštívené 17.03.2018].
URL https://managementmania.com/cs/www-world-wide-web

[20] Bergmann: Jak funguje GPS? [Online; navštívené 05.04.2018].
URL https://www.svethardware.cz/jak-funguje-gps/21826-5

[21] Bondar, I.: TVORBA WWW STRÁNEK :: PROGRAMOVÁNÍ WEBOVÝCH STRÁNEK. [Online; navštívené 28.03.2018].
URL http://www.web-sablony.com/26-165-webtvorba-programovani-internetovych-webovych-stranek-www-na-klic.aspx

[22] Burget, R.: 2. Značkovací jazyky: HTML. [Online; navštívené 27.03.2018].
URL
https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/itw_p02.pdf

[23] Eric Rosebrock, E. F.: *Linux, Apache, MySQL a PHP*. Sybex, Grada, 2004, 2005, ISBN 80-247-1260-1.

[24] Holzner, S.: *JavaScript profesionálně*. Mobil Media a.s., 2003, ISBN 80-86593-40-1.

[25] Hudínek, K.: Operační systémy pro mobilní zařízení, *Bakalářská práce*. [Online; navštívené 27.03.2018].
URL https://is.bivs.cz/th/22099/bivs_b/
Karel_Hudinek_bakalarska_prace_Operacni_systemy_pro_mobilni_zarizeni.pdf

[26] Kosek, J.: *HTML tvorba dokonalých WWW stránek*. Grada, 1998, ISBN 80-7169-608-0.

[27] Kosek, J.: *PHP Tvorba interativních internetových aplikací*. Grada, 1999, ISBN 80-7169-373-1.

[28] Lee, W.-M.: *Beggining Android Tablet Application Development*. Wrox An Imprint of Wiley, 2011, ISBN 978-1-118-10673-0.

[29] Medveď, L.: Vývoj aplikácií pre Apple iOS, *Bakalárska práca*. [Online; navštívené 16.04.2018].
URL https://is.bivs.cz/th/14369/bisk_b/BC__Medved_Lukas_.pdf

[30] Meyer, E.: *Eric Meyer o CSS - ovládněte kaskádové styly*. Zoner software s.r.o., 2004, ISBN 80-86815-03-X.

[31] Motto, T.: Introduction to TypeScript. [Online; navštívené 11.04.2018].
URL https://toddmotto.com/typescript-introduction

[32] Nick Randolph, C. F.: *Windows Phone 7 Application Development*. willey Publishing, Inc., 2011, ISBN 978-0-470-89166-7.

[33] Pošmura, V.: *Apache - Príručka správce WWW serveru*. Computer press, 2002, ISBN 80-7226-696-9.

[34] Radek Hojgr, J. S.: *GPS Praktická uživatelská příručka*. Computer Press a.s., 2007, ISBN 978-80-251-1734-7.

Příloha A

Obsah DVD

- **/src_web** - zdrojové súbory webovej aplikácie (bez jadra frameworku Laravel)
- **/src_mobile** - zdrojové súbory mobilnej aplikácie
- **/doc** - zdojové súbory technickej správy
- **app-release.apk** - inštalačný súbor mobilnej aplikácie
- **database.sql** - skript pre import databázy
- **projekt.pdf** - technická správa
- **README.txt** - súbor obsahujúci informácie pre inštaláciu systému