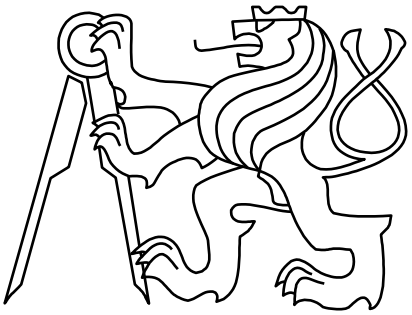
*Na toto místo vložte zadání práce.*

České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačů



Diplomová práce

# Sběr a zpracování dat o provozu automobilů

*Bc. Tomáš Jungman*

Vedoucí práce: Ing. Martin Komárek

Studijní program: Otevřená informatika

Obor: Umělá inteligence

30. prosince 2014

# Poděkování

Rád bych poděkoval vedoucímu práce Ing. Martinu Komárkovy za velkou trpělivost a ochotnou pomoc při tvorbě práce. Děkuji Bc. Romanu Kubů, který mě obětavě zasvětil do tajů světa OBD zařízení. Poděkovat bych také chtěl rodině a přátelům, kteří při mě vždy stáli a podporovali mě během celého studia, a to nejen v duševní rovině, ale také ochotným zapůjčením svých vozů pro účely práce.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 30. prosince 2014 .....

# Abstract

This thesis is aimed towards a design and implementation of application for the Android platform. Application will read and collect data from personal vehicles. It will be based on already existing prototype developed at Metrocar project. Application will also enable upload of collected data to internet server.

# Abstrakt

Cílem této práce je návrh a vývoj aplikace, sloužící pro sběr a záznam dat z osobního automobilu. Aplikace poběží na platformě Android a bude vycházet z existujícího prototypu v projektu Metrocar. Umožní rovněž přenos zaznamenaných údajů na internetový server.

# Obsah

**Obsah** ..... **xi**

**1 Úvod** ..... **1**

1.1 Sdílení vozů (car sharing) ..... 1

1.1 Základní technické překážky ..... 3

1.2 Projekt Metrocar ..... 4

**2 Vymezení cílů práce** ..... **7**

2.1 Android palubní jednotka pro sběr dat ..... 7

2.2 Kompatibilita s různými vozy ..... 8

*2.2.1 Technické aspekty OBD* ..... *8*

*2.2.2 Podpora nestandartních PID kódů* ..... *9*

2.3 Odesílání a zobrazování dat na serveru ..... 10

**3 Analýza stávajícího stavu** ..... **11**

3.1 Stav serveru ..... 11

*3.1.1 Registrace uživatele* ..... *12*

*3.1.2 Vytvoření rezervace* ..... *12*

*3.1.3 Provedení jízdy* ..... *13*

*3.1.4 Záznam o jízdě* ..... *13*

*3.1.5 Nezpracovaná jízdní data* ..... *14*

*3.1.6 Visuální zobrazení dat* ..... *15*

3.2 Stav palubní aplikace ..... 17

*3.2.1 Načtení rezervace* ..... *18*

*3.2.2 Zkušební jízda* ..... *19*

*3.2.3 OBD údaje z vozu* ..... *21*

3.3 Odesílání jízdních záznamů na server ..... 21

**4 Návrh řešení** ..... **25**

4.1 Požadavky na systém ..... 25

*4.1.1 Řešení serverové části ... 26*
*4.1.2 Varianty produktu ... 26*
*4.1.3 Případy použití ... 28*
*4.1.4 Požadavky na palubní aplikaci ... 30*
*4.1.5 Požadavky na serverovou aplikaci ... 34*
4.2 Systém pro sběr jízdních dat ... 38
*4.2.1 Schéma systému ... 38*
*4.2.2 Organizace systému ... 39*
4.3 Palubní aplikace ... 40
*4.3.1 Zapojení palubní jednotky ... 40*
*4.3.2 Návrh komponenty služeb ... 42*
*4.3.3 Návrh GUI ... 43*
*4.3.4 Vývojové prostředí ... 47*
4.4 Serverová aplikace ... 47
*4.4.1 Uživatelské rozhraní ... 47*
*4.4.2 Ukládání dat ... 50*
*4.4.3 API ... 51*
**5 Vývoj palubní jednotky ... 53**
5.1 Vývoj GUI ... 53
*5.1.1 UI Manager a splash screen ... 53*
*5.1.2 Menu a navigace ... 55*
*5.1.3 Nastavení a dialogy ... 57*
5.2 Systém služeb ... 59
*5.2.1 Struktura projektu v IDE ... 59*
*5.2.2 Architektura systému ... 60*
*5.2.3 Sběrnice událostí ... 61*
*5.2.4 Služba OBD ... 63*
*5.2.5 Interpretace PID kódů ... 64*
*5.2.6 Síťová služba ... 67*

*5.2.7 Služba pro ukládání dat* ... *68*
5.3 Využití služeb v GUI ... 68
*5.3.1 Nastavení a kontrolery* ... *69*
*5.3.2 Uživatelské PID kódy* ... *70*
*5.3.3 Přístrojová obrazovka* ... *71*
5.4 Testování ... 72
*5.4.1 Android zařízení* ... *72*
*5.4.2 Vozy* ... *72*
*5.4.3 Testovací případy* ... *73*
*5.4.4 Testovací jízdy* ... *75*
5.5 Zdrojové kódy a dokumentace ... 77
**6 Vývoj serverové aplikace ... 79**
6.1 Vývoj webu ... 79
*6.1.1 Abstrakce databáze* ... *79*
*6.1.2 Adresářová struktura* ... *80*
*6.1.3 API* ... *81*
*6.1.4 Použité JavaScript knihovny* ... *81*
*6.1.5 Přihlašování a úvodní stránka* ... *82*
6.2 Zobrazení dat ... 83
*6.2.1 Vymezení úseku jízdy* ... *85*
*6.2.2 Synchronizace datových bodů* ... *85*
6.3 Další rozšíření ... 87
*6.3.1 GCM* ... *87*
*6.3.2 NFC* ... *88*
*6.3.3 PHP ZIP* ... *89*
6.4 Grafika a uživatelský dojem ... 90
*6.4.1 Zdroje obrázků* ... *90*
*6.4.2 Grafický návrh ikon a pozadí* ... *91*
**7 Výsledky a diskuse ... 93**

7.1 Vyvinutý software ..... 93
7.2 Kritika ..... 93
7.3 Budoucí vývoj ..... 94
*7.3.1 Správa tankování ..... 94*
*7.3.2 Odemykání vozu - lock ..... 94*
*7.3.3 USB NFC čtečka ..... 94*
*7.3.4 Více vozů pod jedním uživatelským účtem ..... 95*
**8 Závěr ..... 97**
**Seznam obrázků ..... 99**
**Seznam literatury ..... 101**
**Seznam použitých zkratek ..... 113**
**Obsah přiloženého CD ..... 115**

# 1 Úvod

## 1.1 Sdílení vozů (car sharing)

Sdílení vozů zažívá od přelomu 21. století velký rozmach [1]. Jedná se o formu autopůjčovny, avšak s tím rozdílem, že primárním cílem provozovatele není hromadění zisku, ale především obecný prospěch. Uživatelé služby si mezi sebe obvykle dělí skutečné náklady na provoz sdíleného vozového parku.

Uživatel takové služby je oprávněn po provedení platné rezervace, využít pro svou potřebu jakékoli dostupné sdílené vozidlo, přičemž obvykle uhradí provozovateli určitou paušální částku, stanovenou dle doby užívání vozu, ujeté vzdálenosti nebo kombinace obojího [20]. Služba je nejčastěji využívána při místních cestách v rámci velkých aglomerací, avšak výjimkou není ani možnost přeshraničních cest (viz. [20]).

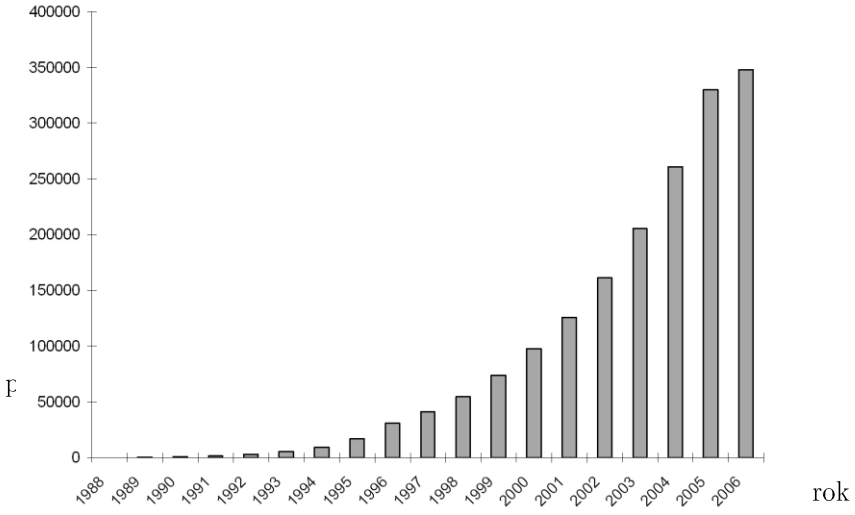
Uživatel tedy může vůz použít, aniž by hradil jeho pořizovací cenu nebo provozní náklady v plné výši, platí jen jejich část, kterou reálně využije. Dle americké souhrnné studie [1] nahradí jeden sdílený vůz v Evropě průměrně 7 vozů v osobním vlastnictví, což pro jejich majitele představuje zhruba 85% úsporu nákladů. V Severní Americe dokonce jeden sdílený vůz nahradí průměrně 15 běžných vozů v osobním vlastnictví, což by odpovídalo 93% ušetřených nákladů.

Finanční úspory však nejsou jediným faktorem, stojícím za úspěchem car-sharingu. Dle [1] může sdílení vozů z různých důvodů ušetřit v průměru minimálně třetinu celkové najeté vzdálenosti. Sdílené flotily se také většinou skládají z úspornějších a novějších vozidel, jejichž efektivita provozu je vyšší. Dle

[2] byla spotřeba průměrného sdíleného vozu o 25% nižší než u vozů v osobním vlastnictví. Oba tyto faktory se positivně odráží na míře znečištění životního prostředí, která je velmi důležitá, především ve velkých městech.

Z tohoto důvodu bývá také častá praxe, kdy města zavádějí parkovací zóny vyhrazené pouze pro sdílené vozy [1]. Parkování je zde obvykle zdarma nebo za sníženou cenu, což dále přispívá k větší popularitě této služby.

***Vývoj počtu uživatelů car-sharingových služeb ve světě***




*Obr. č. 1: Vývoj počtu uživatelů car-sharingových služeb ve světě, převzato z [1]*

## 1.1 Základní technické překážky

Myšlenka sdílení vozů tu nepochybně byla již mnohem dříve, než došlo k masivnímu rozšíření služby na počátku 21. století. Její realizace v praxi však naráží na několik základních technických překážek.

První z nich je sdílení klíčů, tzn., pokud má mít každý uživatel možnost využít kterýkoli z vozů, musí od něj mít klíče (či jiný identifikační prostředek, umožňující vozidlo využít, např. RFID kartu nebo čip).

Dalším problémem je rezervace vozidel, tj. je zapotřebí nějakého mechanismu, který by zajistil, že pokud si zarezervujeme vozidlo na konkrétní den a hodinu, nebude možné, aby vozidlo v tu dobu použil jiný uživatel. V takové situaci by nemusel být na místě k dispozici jiný vůz a my bychom nemohli realizovat cestu.

Třetím poměrně komplexním problémem je výpočet ceny, kterou by uživatel služby za svou jízdu zaplatit. K tomu je zapotřebí měřit nejen dobu, po kterou měl uživatel vozidlo rezervováno, ale také najetou vzdálenost. Záznamy je také potřeba centrálně uchovávat pro potřeby fakturace.

Další překážkou by také mohl být problém s přesunem vozidel mezi jednotlivými stanovišti, v případě že by jich služba měla po městě více. Tato eventualita však není v práci uvažována.

## 1.2 Projekt Metrocar

Metrocar vznikl na ČVUT v Praze jako studentský projekt. Zpočátku projekt pokrýval pouze rezervační aspekt, později se však rozrostl o další technologie a hardwarové prostředky, s ambicí pokrýt celé spektrum car-sharingové problematiky. Cílem projektu je především vyvinout ucelený soubor softwarového řešení pro car-sharingovou společnost, který by řešil technické v (viz výše), jež problematika před vývojáře staví.

Takové řešení by v budoucnu mělo umožnit např. neziskovým organizacím či občanským sdružením rozběhnout vlastní službu sdílení vozidel.

Historie projektu sahá do roku 2009 [3,4] kdy jako první krok vzniká online aplikace, obstarávající především reservační aspekt car-sharingové problematiky.

Pro zbývající dva aspekty, tedy sdílení klíčů a identifikaci byl později v témže roce [3] vytvořen hardwarový prostředek pro umístění do vozu, který měl za úkol především sledovat trasu pohybu vozu (pomocí GPS) a zajišťovat autorizaci uživatele (pomocí RFID), avšak nebyl nijak spojen se serverovou rezervační částí.

S ohledem na vysokou pořizovací cenu tohoto prostředku bylo později rozhodnuto, že zařízením, umísťovaným do vozu, tzv. „palubní jednotkou“, bude mobilní telefon, či tablet s operačním systémem Android [3,5,21].

V roce akademickém 2012/2013 vzniká první experimentální Android aplikace[3], která kromě zjišťování polohy z GPS a spojení se serverovou částí umožnuje také přímé spojení s vozem pomocí standardu OBD2 [22].

Do serverové části aplikace jsou mezitím přidány další moduly, jako např. účetní modul, umožňující převod reálných finančních prostředků na účet uživatele [6] nebo geografický modul pro zobrazení jízd na mapě [7].

V době zahájení příprav této práce byla serverová část nasazena v testovacím režimu online na serveru autonapul.cz [23]. Zdrojové kódy byly k dispozici v repositáři na projektovém serveru Assembla [24].

# 2 Vymezení cílů práce

## 2.1 Android palubní jednotka pro sběr dat

Tato práce pokračuje v úsilí o vývoj software pro palubní jednotky na platformě Android [21], v návaznosti na experimentální Android aplikaci z roku 2013 [3]. Ta je určena především pro ověření konceptu (proof of concept) použití dostupného přenosnému Android zařízení pro sběr jízdních dat a jejich odeslání na server. Nicméně vzhledem ke své experimentální povaze není ve své podobě vhodná k reálnému nasazení do vozů carsharingové služby.

Důležitou součástí práce je analýza a identifikace klíčových aspektů palubního software, které by umožnili jeho nasazení a praktické každodenní použití v reálných vozech. Shromážděné poznatky jsou následně aplikovány při vývoji Android aplikace, jež bude hlavním produktem práce.

Mezi klíčové aspekty, které je třeba analyzovat patří např. uživatelská přívětivost aplikace, jež v [3] není zpracována. Dále způsob připojení a práce s OBD2 adaptérem pro sběr informací z vozu. Poměrně komplexní a pro práci stěžejní oblastí je také výběr a zpracování jízdních dat, ať už pocházejících z OBD zařízení nebo z jiných sensorů či videokamery. Pozornost si žádá též způsob odesílání zaznamenaných údajů na server, jež se ukázal být v prototypu problematický [3].

## 2.2 Kompatibilita s různými vozy

### *2.2.1 Technické aspekty OBD*

Prototyp palubní Android aplikace [3] umožňuje čtení jízdních údajů přímo z vozu. Využívá k tomu OBD2 [22] zařízení s čipem ELM327 1.5 [28,32], které se s Android zařízením spojí pomocí bezdrátové technologie Bluetooth [29].

OBD zařízení se pomocí CAN [30] diagnostické sběrnice připojí k centrální řídící jednotce (ECU [31]) vozu, od níž je schopno získávat aktuální data z různých sensorů automobilu [28]. Čip ELM327 podporuje všechny protokoly, jež jsou součástí standartu OBD2 [22,32], což prakticky zajišťuje kompatibilitu se všemi vozy disponující CAN sběrnicí. Přítomnost CAN sběrnice i částečná podpora OBD standartu jsou také vyžadovány zákonem na hlavních světových automobilových trzích, jako jsou USA, Čína a Evropská unie [22].

Ačkoli na tomto poli panuje shoda, co se týče sběrnice a komunikačních protokolů, kompatibilita samotných zpráv (PID kódů [28]) se omezuje pouze na oblast měření emisí, vyžadovanou zákonem [22] u nových vozů. Výrobci automobilů tak v praxi používají namísto standartních PID kódů definovaných OBD standardem, vlastní proprietární kódy. Ty se ještě navíc liší v závislosti na konkrétním typu ECU, kterých existuje značné množství. U starších vozů disponujících CAN sběrnicí potom může podpora standardních OBD PID kódů chybět zcela.

Nastává tedy situaci, kdy sice ECU vozu dokáže nabídnout mnoho užitečných informací dalece přesahující specifikace standardu OBD, ale my je kvůli proprietární a nejednotné povaze PID kódů nejsme schopni vyžádat.

Americká nezisková organizace ETI [33] se tento problém pokouší řešit cestou shromažďování technických specifikací od všech hlavních výrobců automobilů. Pro přístup k informacím, je ovšem vyžadováno placené členství s cenou začínající na 5 tis. amerických dolarech p.a., ovšem toto může být jediný oficiální způsob, jak získat přístup k ověřeným nestandartním PID kódům[22].

### 2.2.2 Podpora nestandartních PID kódů

Stávající přístup prototypu [3] počítá pouze s vozy plně podporující standartní OBD2 [22] PID [28] kódy. To by však vedlo k několika problémům. Nemohli bychom používat starší vozy, které sice mají výkonnou ECU, avšak nepodporují standartní kódy. U novějších vozů bychom byli omezeni pouze na několik málo jízdních údajů. A v budoucnu, by bylo obtížnější se přizpůsobit, pokud by došlo k rozšíření nového sjednocujícího standardu napříč výrobci.

Z těchto důvodů je vhodné problematiku analyzovat a navrhnout universální řešení, které by umožnilo použití proprietárních kódů k získání jízdních dat. Navržené řešení je také nutno implementovat a pro ověření funkčnosti otestovat s reálným vozem, bez podpory standartních OBD PID kódů.

## 2.3 Odesílání a zobrazování dat na serveru

Jak již bylo řešeno, palubní Android aplikace slouží ke sběru dat z jízdy. Ta následně odesílá na server k uložení a zpracování. Tím muže být např. výpočet najeté vzdálenosti a vyúčtování jízdy [6], uzavření rezervace apod.

Data však mohou sloužit i k přímému visuálnímu zobrazení, lze tak např. zobrazit trasu jízdy na mapě, visuálně analyzovat naměřené údaje na grafech [7] nebo dát uživateli k dispozici mapu s aktuální polohou sdílených vozidel k dispozici.

Pro účely této práce je žádoucí, aby všechny komponenty umožňující odeslání dat na server byly plně funkční a přístupné online pro použití palubní jednotkou. Stejně jako komponenty pro visuální zobrazení dat, jež poslouží ke kontrole naměřených údajů.

V předcházejících pracích v projektu Metrocar [3,4,7] bylo předestřeno, že v serverové části dochází k určitým problémům spojeným s velkým množstvím zaznamenaných údajů.

Cílem práce je proto v tomto bodě je otestovat funkčnost serveru týkající se jízdních záznamů a chování serveru při běžném zatížení. Případné nalezené nedostatky analyzovat, navrhnout a následně také implementovat a otestovat řešení, které by odhalené nedostatky řešilo.

S ohledem na použití poměrně exotického programovacího jazyka Python [34] a frameworku Django [35] a jejich neznalost autorem této práce je třeba velmi pečlivě zvážit rozsah a formu provedených úprav.

# 3 Analýza stávajícího stavu

Tato kapitola odpovídá první iteraci iterativního přístupu k této práci. Jejím účelem je prozkoumání aktuálního stavu systému serverové a palubní části projektu Metrocar, pro účely pozdějšího návrhu vylepšení palubní jednotky.

## 3.1 Stav serveru

Serverová část projektu Metrocar je v době analýzy nasazena na serverech hostingové společnosti Roští.cz [25]. Zde je rozdělen na dvě samostatné části, sdílející jednu databázi typu PostreSQL [26]. První část s názvem „mfe“ představuje uživatelskou část přístupnou řadovým uživatelům, obecně označovanou také jako frontend, dostupnou z URL „http://www.autonapul.cz/“. Druhá část s názvem „metrocar“ představuje administraci pro správce služby (backend), přístupnou z URL „http://admin.autonapul.cz“.

Přístup k administraci PostgreSQL databáze je možný ze serveru „https://pg.rosti.cz/“ po zvolení databáze „metrocar“ a zadání příslušných přístupových údajů.

### 3.1.1 Registrace uživatele

V rámci testování bylo nejprve nutné na serveru vytvořit uživatelský účet. Pro tyto účely bylo využito existujícího registračního formuláře na „http://autonapul.cz/registrace/“.

V administrační části byla registrace schválena a do databáze bylo přidáno nové testovací vozidlo s parametry reálného vozu autora této práce. Byl také vytvořen klíč pro identifikaci palubní jednotky, který je potřeba zaslání jízdních dat na server.

Zde lze pouze podotknou, že vše fungovalo bezchybně, vč. validace údajů a zaslání registračního emailu. Na vytvořený uživatelský účet se dalo přihlásit a zobrazené údaje byly v pořádku.

### 3.1.2 Vytvoření rezervace

V uživatelské části, po přihlášení na vytvořený účet byla vytvořena rezervace předem vloženého vozu. Vytvoření proběhlo v pořádku, rezervace se objevila mezi čekajícími rezervacemi. Vytvoření rezervačního záznamu bylo též ověřeno v administraci databáze, kde byl nový záznam podle očekávání nalezen.

### 3.1.3 Provedení jízdy

Po vytvoření rezervace byla provedena zkušební jízda reálným vozem, při které byla použita experimentální Android aplikace [3] v mobilním telefonu jako palubní jednotka (viz kapitola **Chyba! Nenalezen zdroj odkazů.**).

Během jízdy palubní jednotka vytvořila záznam jízdních dat a polohy GPS. Tento záznam se aplikace pokusila po skončení jízdy odeslat na server, avšak s neznámým výsledkem.

Na straně serveru byla bezprostředně po jízdě ověřena přítomnost nových záznamů. V uživatelské části byl nejprve ověřen stav příslušné rezervace, stahující se k aktuální jízdě. Stav rezervace byl nadále označen jako „čekající“, tento byl znovu ověřen s odstupem 15 minut, 1 hodiny a 1 a 5 dnů od provedení jízdy, avšak stav jízdy zůstával i nadále beze změny, což lze označit jako problém neboť tento stav neodráží skutečnost.

### 3.1.4 Záznam o jízdě

Po provedení zkušební jízdy nenastala z pohledu uživatele v systému žádná změna. Je proto nutné ověřit v administraci serveru, zda o jízdě existuje nějaký záznam.

Nejprve byla přítomnost nového záznamu o jízdě ověřena v sekci „Cars“ pod položkou „Jízdy“. Přestože sekce obsahovala množství záznamů, z roku 2012 a trojici záznamů z roku 2013, žádný nový záznam nebyl nalezen. V administraci

databáze byla zkontrolována tabulka „cars_journey“, jež slouží jako zdroj dat pro zmíněnou sekci, ani zde nebyl přítomen nový záznam.

Tento fakt lze rovněž označit za problém, neboť tento záznam má sloužit jako základ pro vyúčtování jízdy a jeho neexistence prakticky znemožňuje provedení úhrady za jízdu.

### 3.1.5 Nezpracovaná jízdní data

Ačkoli na serveru nebyl přítomen shrnující záznam o jízdě, je možné, že jsou přítomna alespoň jízdní data v nezpracované podobě. Jejich přítomnost byla zkoumána nejprve v administraci, v sekci „Audit“ pod položkou „Usage_history“. Zde bylo zvoleno datum „od“ a datum „do“ odpovídající reálnému času provedení jízdy, a vybráno příslušné vozidlo, avšak nedošlo k zobrazení žádných záznamů, jak na přítomné OSM[36] mapě, tak ani jinde na stránce.

V administraci databáze jsou jízdní záznamy uloženy v tabulce „car_unit_api_logentry“. Přesto, že tabulka již obsahovala množství záznamů, nebyly nalezeny žádné nové, které by časovou známkou (timestamp) odpovídali době zkušební jízdy.

Lze tedy konstatovat, že nedochází k uložení jízdních záznamů, pocházejících z palubní jednotky, do databáze serveru, což způsobuje celkovou nefunkčnost komponent pro správu a zobrazení jízd a jízdních dat.

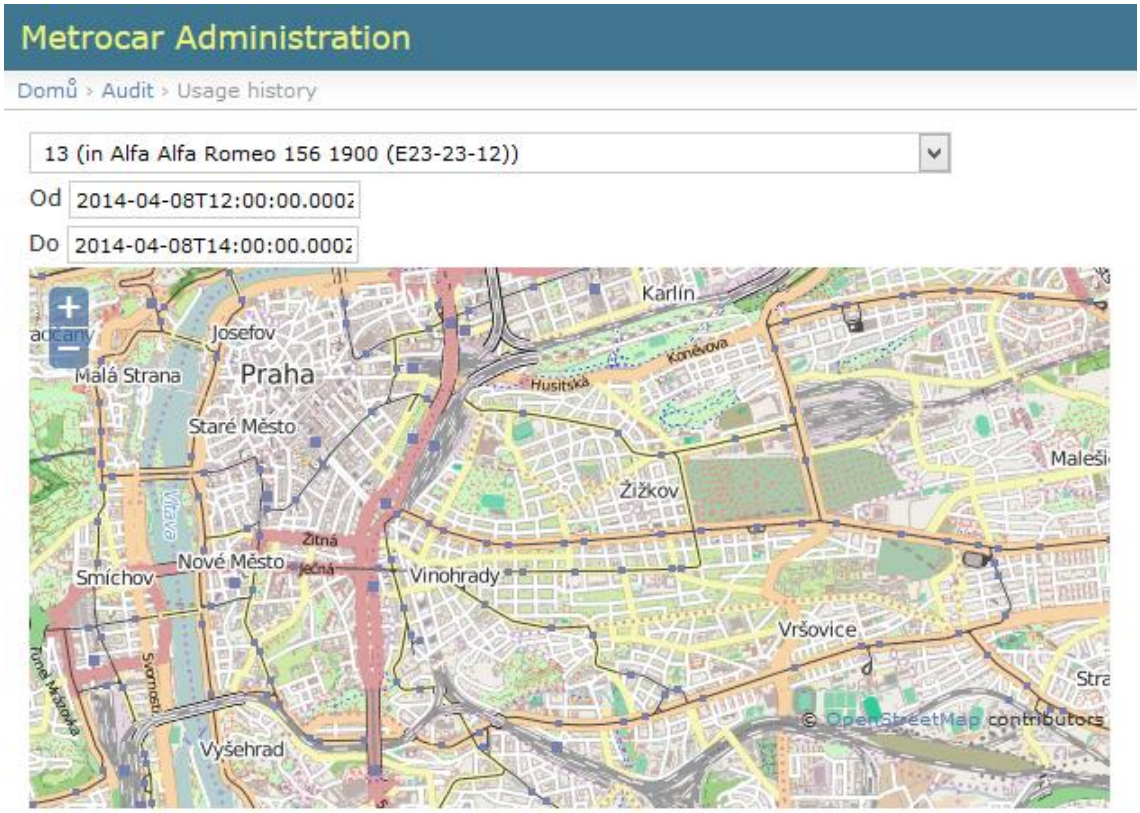
### 3.1.6 Visuální zobrazení dat

Ačkoli na serveru nedošlo k zaznamenání dat z testovací jízdy, bylo zjištěno, že databáze obsahuje dostatek jiných dat. Tato lze využít k ověření funkčnosti geografické komponenty [7](OSM mapy) a její schopnosti zobrazit jízdní data v grafech.

V databázi jízdních záznamů bylo nalezeno více než 100 tisíc záznamů, které byly opatřeny časovými razítky, na jejichž základě bylo poté vyžádáno zobrazení dat v GUI administrace.

I přes ověřenou přítomnost záznamů, nedošlo v GUI k zobrazení záznamů u vozů, k nimž příslušeli záznamy v databázi dle parametru „unit_id“. Stav GUI je zachycen na obrázku (Obr. č. 2), kde po zadání příslušných hodnot a výběru odpovídajícího vozu z nabídky, nedochází k žádné změně.

Požadavek na zobrazení dat byl testován v prohlížečích Firefox, Chrome a Internet Explorer na OS Windows 8 64bit, a Firefox a Chromium pro OS Linux (Ubuntu 11.1). Prohlížeče byli k datu provedení testu (12. 4. 2014) aktualizované a ve svých posledních dostupných vydáních.




*Obr. č. 2: Původní GUI pro zobrazení jízdních záznamů*

Stav GUI zůstával vždy beze změny, pouze se zobrazenou mapou. Vzhledem a absenci potvrzovacího tlačítka, které by iniciovalo zobrazení dat, byla provedena také kontrola odeslání formuláře pomocí Firefox pluginu Firebug a Nástroje pro vývojáře prohlížeče Chrome. Bylo zjištěno, že v odeslání je aktivováno volbou vozu z vysouvacího menu.

```
http://admin.autonapul.cz/audit/usage_history/query/

{
  "status": "ok",
  "results": [],
  "request_id": 19
}
```

*Obr. č. 3: Odpověď serveru na dotaz pro zobrazení jízdních dat*

Dle zjištění dochází k odeslání formuláře, avšak server nevrací v odpovědi žádná data i přesto, že jsou záznamy v databázi k dispozici. Lze tedy konstatovat, že zobrazení jízdních záznamů není v současné době na serveru možné.

## 3.2 Stav palubní aplikace

Zdrojové kódy palubní aplikace byly dodány autorem prototypu [3], aplikace byla nakonfigurována a sestavena v IDE ADT [37]. Sestavení proběhlo v pořádku a došlo ke spuštění aplikace na fyzickém Android zařízení. Dalším krokem k ověření funkce palubní aplikace je provedení zkušební jízdy reálným vozem.

Aplikace je spolu s OBD2 Bluetooth diagnostickým zařízením [28] nasazena ve voze autora této práce. Hostitelským zařízením pro aplikaci je mobilní telefon typu „HTC One S“ s OS Android verze 4.1.1 a připojením k internetu prostřednictvím 3G. Android telefon a OBD adaptér jsou spárovány v rámci technologie Bluetooth [29].

Během zkušební jízdy je aplikace v režimu „debug“ stále fyzicky propojena s osobním počítačem a IDE ADT přes USB kabel, za účelem záznamu logu a analýzy síťové komunikace.

### 3.2.1 Načtení rezervace

Před jízdou bylo nutné v aplikaci nastavit přístupový klíč palubního zařízený vytvořený v administraci serverové části, který bude sloužit k identifikaci palubní jednotky.

Dalším krokem bylo přihlášení uživatele přihlašovacím jménem a heslem. Po stisku příslušného tlačítka došlo v aplikaci k načtení rezervací, což sice nebylo nikterak visuálně potvrzeno uživateli, záznam o proběhlé síťové komunikaci IDE ADT [37] však ukazoval na fakt, že byly načteny všechny aktivní rezervace vztahující se k vozidlu, a to ve formátu JSON [60].

Aplikace v případě existující platné rezervace povolila zahájení jízdy. Testován byl také scénář, kdy neexistovala pro vůz evidovaný aplikací žádná platná registrace, což aplikace reflektovala a neumožnila zahájení jízdy.

Lze konstatovat, že stažení rezervací API serveru a jejich interpretace funguje dle očekávání.

### 3.2.2 Zkušební jízda

Po přihlášení do aplikace a načtení platné rezervace, již dříve vytvořené na serveru (viz kapitola 3.1.2) byla provedena zhruba 5 minutová zkušební jízda reálným vozem.

Během jízdy docházelo průběžně ke změně parametru „Engine RPM“ symbolizující otáčky motoru na hodnotu zhruba čtyřnásobně menší nežli byly skutečné otážky, ostatní parametry získáváné z vozu měli konstatní hodnotu 0, s vyjímkou rychlosti “Speed”, jež se po dosažení reálné rychlosti okolo 10 km/h na tachometru vozu, změnila na hodnotu 1.

Poloha vozu symbolizovaná GPS souřadnicemi v poli “GPS” se během jízdy průběžne měnila. Získané souřadnice byly zpětně ověřeny po zkončení jízdy a přibližně odpovídali trase jízdy.

Pole state, symbolizující stav spojení Android aplikace s OBD diagnostickou jednotkou zobrazoval po celou dobu jízdy hodnotu “Connected”, což je dle [3] správný stav.

*Obr. č. 4: Jízdní obrazovka prototypu Android aplikace, převzato z [3]*

Po ukončení jízdy bylo patrné visuální "zamrznutí" aplikace. Údaje získané zpětně z ADT naznačují, že k tomu došlo v momentě kdy aplikace odesílala nasbírané jízdní údaje z interní databáze na server. To bylo ověřeno také v záznamu síťové komunikace v ADT.

Formát odesílaných dat byl ověřen vůči specifikaci serverového API pro příjem jízdích záznamů [3,7] a byl shodný. Přesto nedošlo k uložení záznamů do databáze na straně serveru (viz 3.1.5).

V tomto bodě lze konstatovat, že prototyp palubní aplikace plní svou funkci dobře v oblasti záznamu dat, je však třeba odhalit příčinu bránící uložení dat na server. Bližší analýzu si zaslouží také zmíněná nepřesnost či absence jízdních údajů získaných z OBD a výkon aplikace.

### 3.2.3 OBD údaje z vozu

Během zkušební jízdy bylo zjištěno, že jízdní údaje získávané z ECU vozu přes OBD adaptér nejsou buď vůbec k dispozici, nebo jsou nepřesné.

Zdrojový kód prototypu [3] byl proto modifikován tak, aby bylo možné z vývojového (debug) logu přečíst veškeré PID dotazy odesílané do vozu a odpovědi na ně.

Během další testů s reálným vozem, bylo zjištěno, že ECU vozu odpovídá na většinu dotazů kódem „7F 09 12“, což znamená dle [8] ekvivalent chybové řetězce. Vzhledem k tomu, že ostatní parametry a celkový průběh komunikace probíhá dle očekávání, je pravděpodobnou příčinou chyby skutečnost, že vůz nepodporuje standartní PID kódy, jak již bylo diskutováno (viz kapitola 2.2).

Výsledky testu prototypu s jiným vozem potom tuto hypotézu potvrzují, neboť jsou správně zobrazeny hodnoty aktuální rychlosti, otáček motoru a teploty motoru. Na dotaz na pozici plynového pedálu („throttle“) odpovídá ECU vozu rovněž kódem „7F 09 12“, což naznačuje, že nepodporuje příslušný PID kód.

## 3.3 Odesílání jízdních záznamů na server

Jak již bylo zmíněno, palubní aplikaci se po provedení zkušební jízdy nedařilo odeslat data na server. V tomto bodě je úkolem zjistit proč tomu tak je.

Pomocí ADT a mírných úprav zdrojového kódu byla získána jízdní data ve formátu JSON, které se aplikace pokoušela odeslat.

Abychom nemuseli testovací jízdu neustále opakovat, je pro simulaci dotazu vytvořena v jazyce JAVA [75] v IDE Netbeans [38] testovací utilita s názvem „AjaxTester“ (zdrojové kódy k dispozici na přiloženém CD). Jejím úkolem je simulovat odeslání dat palubní aplikací, abychom mohli sledovat odpověď serveru.

Použitím simulační utility bylo zjištěno, že server na veškeré dotazy obsahující data z testovací jízdy odpovídá pouze HTTP kódem 400, aniž by odesílal nějaké informace v těle odpovědi, který obvykle značí chybnou formu dotazu. Stejně odpovídá také na zkrácenou podobnu dat obsahující pouze jednotky záznamů. Při použití referenčního dotazu [7] je odpověď stejná, ačkoli by dle specifikace [7] měla obsahovat informace o výsledku či povaze chyby.

Pro zjištění bližších informací o příčinách chyby byl přes FTP načten chybový log serveru z adresáře „/logs/error.log“, kde byl zaznamenán vysoký počet výskytů chybových hlášení při přístupu k URL náležící k API pro komunikaci s palubní jednotkou.

Typický záznam logu v souvislosti se zmíněnou chybou je zachycen na následujícím obrázku (Obr. č. 5).

```
2014/04/22 14:34:19 [error] 651#0: *51957 readv() failed
(104: Connection reset by peer) while reading upstream,
client: ::ffff:89.187.152.40, server: admin.autonapul.cz,
request: "POST /api/log/ HTTP/1.0", upstream:
"uwsgi://unix:///home/apps/app_00008/app.sock:", host:
"admin.autonapul.cz"
```

*Obr. č. 5: Ukázka záznamu chyby z chybového logu serveru*

Vzhledem ke skutečnosti, že autor práce nedisponuje znalostí serverových platforem Python [34] a Django [35] potřebné k opravě nalezených chyb, byli kontaktování původní autoři serverové částí [6,7] s žádostí o radu. Z komunikace ovšem brzy vylynulo, že si problém vyžádá širší lokální testování, což by již vyžadovalo poměrně extensivní znalost serverových platforem.

Závěrem v tomto bodě je tedy konstatování, že serverová aplikace nemůže být ve svém stavu využita pro uložení ani zobrazení jízdních záznamů. Současně není ani v silách autora této práce stav serveru napravit.

# 4 Návrh řešení

Tato kapitola je druhou iterací iterativního přístupu k této práci. Hlavním úkolem je nabídnou možná východiska a odpovědi na otázky, jež vyvstanuly během předchozí analytické iterace. A to především ve formě obecné softwarového návrhu jednotlivých součástí systému pro sběr jízdních dat a specifikací požadavků, jež mají splňovat.

## 4.1 Požadavky na systém

Problematika sběru a zpracování jízdních dat si žádá použití systému různých spolupracujících komponent. Těmi hlavními jsou palubní jednotka a internetový server, ovšem v závislosti na účelu použití může být zapotřebí uvažovat i další komponenty (např. RFID čtečku, OBD adaptér, webkameru apod.), které musí být v návrhu zohledněny.

Je potřeba stanovit nejčastější případy použití systému a podle nich v systému identifikovat nejdůležitější komponenty, kterým je třeba se věnovat, stejně jako problémům a slabým místům, odhaleným během analýzy stávajícího řešení (v kapitole 3).

### 4.1.1 Řešení serverové části

Dle provedené analýzy nemůže být pro účely této práce využita stávající serverová aplikace projektu Metrocar. Bez serverové aplikace se však obejít nelze, neboť slouží k centrální evidenci zaznamenaných jízd ze všech zainteresovaných palubních jednotek. Slouží také k zobrazení zaznamenaných geografických a temporálních dat, jež by bylo v palubní jednotce obtížně realizovatelné.

Je proto zřejmé, že musí být vytvořena zcela serverová aplikace se základní funkčností, jež by alespoň částečně nahradila stávající server. Zároveň musí být specifikováno API pro komunikaci mezi palubní jednotkou a novým serverem. API musí být navrženo s ohledem na možnosti stávajícího serveru, tak aby je zde mohli pozdější přispěvatelé do projektu Metrocar implementovat.

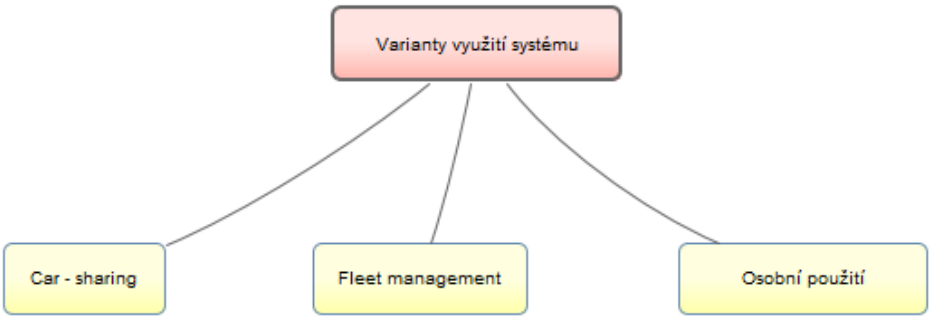
### 4.1.2 Varianty produktu

Potřeba nové serverové aplikace do projektu vnesla určitou míru volnosti týkající se možností využití aplikace. Byla vyvolána diskuse s původním autorem prototypu palubní jednotky [3] a vedoucím práce.

První možnost využití systému sběru dat pro car-sharing je již dána. Objevují se však také návrhy použití systému pro „fleet management“, zprávu firemní flotily vozidel, tedy jako formu elektronické knihy jízd. Dalším zajímavým směrem, kterým by se mohl projekt rozvíjet je osobní využití, kde by byl cílovou skupinou řidič vlastnící osobní vozidlo. Ten by mohl systém využít nejen jako černou skříňku pro případ nehody, ale také pro evidenci a správu jízd. Tato

osobní verze by také mohla sloužit např. v autoškolách či závodních vozech pro analýzu jízdních údajů a videozáznamu začínajících i profesionálních řidičů.

Členění variant je zachyceno na následující jednoduché mapce (Obr. č. 6).:

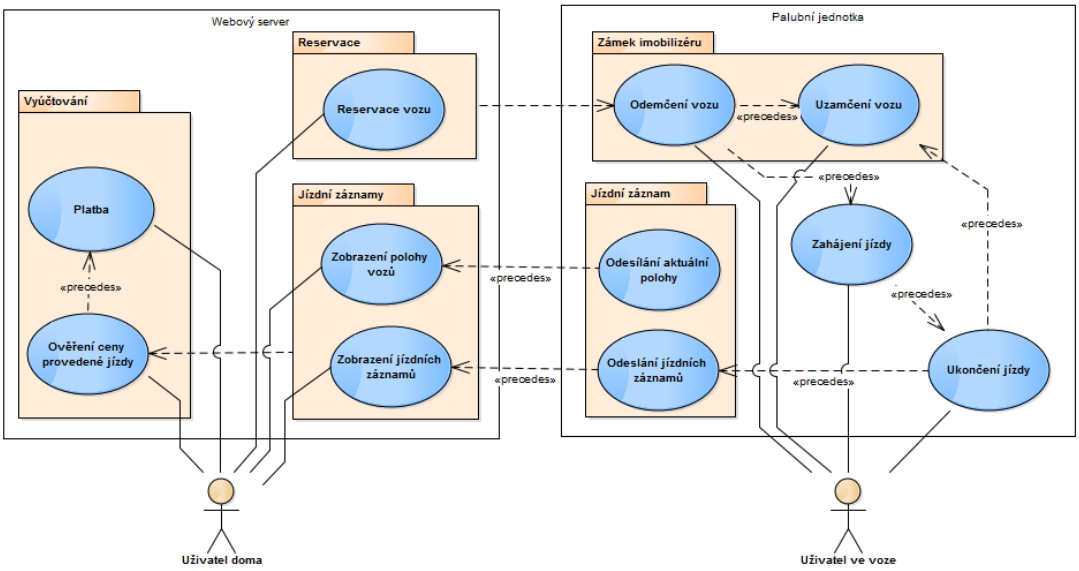



*Obr. č. 6: Mapka variant systému dle využití*

Ačkoli tato práce nemůže plně postihnout aspekty všech tří uvedených směrů najednou, lze pomocí analýzy případů použití identifikovat jejich společné znaky a ty následně zohlednit v softwarovém návrhu.

### 4.1.3 Případy použití

Vzhledem k tomu, že car-sharingová služba má ze všech předestřených variant nejkomplexnější model použití, byla vybrána pro sestavení diagramu (viz Obr. č. 7). Ten zachycuje nejdůležitější činnost uživatelů v systému.




*Obr. č. 7: Use case diagram, případy použití systému*

Z diagramu je jasné, že jak serverovou, tak i palubní část lze rozdělit na několik logických celků.

Aby uživatel mohl provést jízdu, nejprve musí přes internet vytvořit v serverové části reservaci vozu, ta se nesmí překrývat s reservací jiného uživatele na stejný vůz (apod.), což v modelu obstarává logická reservační část.

Palubní jednotka poté umožní odemčení vozu na základě platné reservace, což obstarává logická část „zámek imobilizéru“. Uživatel následně svými akcemi iniciuje začátek a konec jízdy. V jejím průběhu odesílá palubní jednotka aktuální informace o poloze a další data, po skončení potom informuje server, případně na něj odešle zbylé relevantní informace. To je úkol logické části palubní jednotky s názvem „Jízdní záznam“.

Na serveru je pak možné zobrazení aktuálních jízdních dat uživateli, především polohy vozů běžnými uživateli, případně i ostatních údajů pro administrátora systému. Tuto funkcionalitu obstarává logický celek „Jízdní záznamy“, jež přijímá data z příslušné logické části palubní jednotky.

Stejná logická část se stará také o uložení a zobrazení ucelených jízdních záznamů, které později budou sloužit např. pro fakturaci v car-sharingové variantě, účetní účely ve variantě pro fleet-management či pro běžnou osobní evidenci ve variantě pro osobní použití.

Logická část “Vyúčtování” představuje abstrakci účetní komponenty, jež byla v původním projektu implementována [6]. Zde má uživatel v “na základě provedených jízd možnost provést úhradu nákladů na jízdu. Fungování této komponenty by však bylo odlišné ve variantě Fleet management by však bylo odlšn, neboť firemní uživatel typycky nebude provádět přímou platbu za každou jízdu. V osobní variantě by potom tato komponenta chyběla úplně, neboť cílovým uživatelem je zde jednotlivý řidič, který neprovádí žádnou úhradu ani vyúčtování nákladů třetí osobě.

Podobně je na tom i komponenta pro práci s imobilizérem, který je klíčovou součástí pro variantu Car-sharing. Pro firemní a osobní variantu však

není jeho využití nutné, neboť zde není vyžadované (a z hlediska bezpečnosti ani vhodné) aby byl přístup do sledovaného vozidla umožněn více osobám. Ve variantě pro osobní užití navíc není ani předpokládána trvalá instalace palubní jednotky ve vozidle.

V tomto místě lze shrnout, že analýzou diagramu použití systému byly identifikovány klíčové logické součásti systému. Těmi byly součásti obstarvající jízdní záznamy, a to na straně serveru i palubní jednotky, neboť byly použity ve shodné podobě napříč všemi variantami produktu.

Další návrh bude tedy pokračovat primárně pro osobní variantu produktu, neboť tuto lze jako jedinou plně realizovat pouze za použití společných logický součástí. Pro ostatní varianty budou navrženy pouze služby na straně palubní jednotky, pro podporu specifických technologií (např. NFC), jež nebudou nemusí mít v systému uživatelskou funkci.

### 4.1.4 Požadavky na palubní aplikaci

Palubní Android aplikace je v systému určena primárně ke sběru jízdních dat. Aplikaci bude vždy hostit zařízení s OS Android, které má být dle specifikace [39] vybaveno různými sensory, mezi které patří GPS sensor polohy, akcelerometr, proximity sensor [40], budeme také předpokládat přítomnost zabudované videokamery. Z oblasti konektivity budeme předpokládat dostupnost technologie Bluetooth, a dále Wi-Fi [41] a 3G [42] modulů pro připojení k internetu.

Uvedené sensory můžeme použít jako sekundární prostředky pro sběr dat, vedle primárního zdroje, jímž je OBD adaptér.

#### *4.1.4.1 Funkční požadavky*

Požadavky pro sběr dat z OBD:

- Navázání a udržení spojení s OBD Bluetooth adaptérem
- Odesílání, příjem a interpretace libovolných PID zpráv z OBD
- Možnost uživatelské definice PID zpráv, jejich interpretace a povelení či zakázání přímo v GUI aplikace
- Ukládání dat z OBD do interní paměti zařízení

Požadavky pro sběr dat z dalších sensorů:

- Aktivace a příjem dat z GPS sensoru, záznam do paměti
- Využití akcelerometru pro měření přetížení, záznam do paměti
- Využití zabudované videokamery pro záznam jízdy do souborů
- Umožnění ukládání videozáznamu na externí paměťovou kartu

Požadavky na síťovou komunikaci:

- Možnost přihlášení uživatele, za asistence serveru
- Možnost stažení standartních PID kódu pro OBD ze serveru
- Schopnost průběžně odesílat jízdní data na server během jízdy
- Schopnost odeslat na server celkový záznam o jízdě

Požadavky na GUI:

- Přihlašovací obrazovka
- Obrazovka s jízdními daty
- Uživatelské potvrzující dialogy, před aktivací systémových funkcí
- Zobrazení průběžného náhledu zachycovaného videozáznamu
- Zobrazení vybraných jízdních dat z OBD
- Zobrazení polohy vozu ve vhodné formě

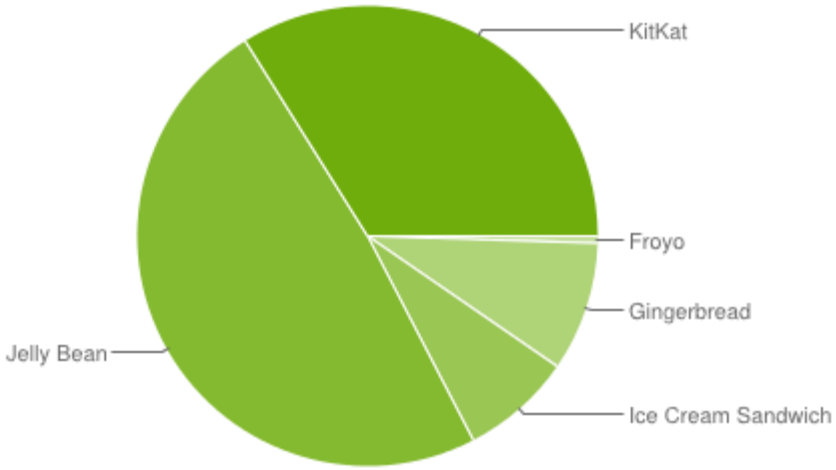
Další požadavky:

- Načtení identifikace z NFC karty
- Možnost spojení s USB videokamerou

#### *4.1.4.2 Nefunkční požadavky – podpora OS*

V tomto bodě bylo potřeba učinit rozhodnutí, jaké úrovně API (Android API Level [43]) bude aplikace podporovat, což v praxi ovlivní verzi podporovaného OS a tím pádem také počet zařízení, s nimiž bude aplikace kompatibilní.

Na grafu (Obr. č. 8) je znázorněno zastoupení jednotlivých verzí OS Android, dle firmy Google [44] (aktualizováno k 1. 12. 2014). Zde je patrné že majoritní podíl mají verze „Jelly Bean“ (4.1.x – 4.3), představující úrovně API (16 – 18) a „KitKat“ (4.4.x), s úrovněmi API (19 a 20). Dohromady jsou tyto verze zastoupeny na více než 80% cílového trhu, jedná se také o dvě nejaktuálnější verze OS, které jsou si velmi podobné. Jejich podpora je proto nutností a bude vyžadovat pouze minimální režii.




*Obr. č. 8: Graf zastoupení verzí OS Android na trhu, zdroj [44]*

Otázkou pak může být podpora verze „Ice Cream Sandwich“ (4.0.x), která má podíl přibližně 8%, avšak její podpora by si vyžádala určitou režii a intenzivnější testování. Vzhledem k tomu, a k faktu, že autor práce nemá k dispozici fyzické zařízení s tímto OS, bude vhodnější toto verzi nepodporovat.

Zcela mimo možnost podpory potom stojí verze „Gingerbread“ (2.3.x) a „Froyo“ (2.2), jejichž architektura je velmi rozdílná. Vzhledem k jejich stáří a souvisejícímu stáří hardware zařízení, na nichž jsou obvykle instalovány, zde obvykle chybí i nezbytné sensory a technologie, jež jsou pro palubní jednotku nezbytné.

Požadavky na podporu OS Android:

- Podpora API Level 16 – 20, odpovídající verzi 4.1.x – 4.4.4

#### 4.1.4.3 Nefunkční požadavky – ostatní

Prototyp palubní aplikace [3] je členěn do dvou částí, první je část obstarávající interakci uživatele (GUI) a druhá potom obstarává ostatní služby (tzv. „engine“). Toto členění je vhodné zachovat z důvodu pozdější úprav GUI pro nasazení v různých verzích systému (viz kapitola 4.1.2).

Vzhledem k faktu, že již původní aplikace měla příležitostné problémy s výkonem (viz kapitola 3.2.2) a velkému počtu funkčních požadavků je vhodné použít vícevláknový přístup (multi-threading). Tento je vhodný též s ohledem na aktuální trend v oblasti hardware, kdy u přístrojů s OS Android dochází k velkému rozšíření vícejádrových procesorů.

Ostatní nefunkční požadavky:

- Zachování členění aplikace na GUI část a knihovní část „engine“, poskytující služby
- Použití vícevláknového přístupu

### 4.1.5 Požadavky na serverovou aplikaci

Serverová část systému má za úkol především evidovat uživatele, provedené jízdy, ukládat podrobná jízdní data a umožnit jejich zobrazení.

Server bude dostupný online, pod předem známou URL adresou. Bude nabízett uživatelský interface (GUI) pro použití v běžném webovém prohlížeči a API pro palubí jednotky, kam budou tyto zasílat svá zaznamenaná data.

#### 4.1.5.1 *Funkční požadavky*

Požadavky na API:

- Uložení aktuálních dat z jízdy (smazání neaktuálních)
- Uložení celkového záznamu jízdních dat
- Možnost přihlášení uživatele do palubní jednotky
- Možnost vyžádání standardních PID kódů a vzorců

Požadavky na záznam dat:

- Kompresní mechanismus pro snížení objemu ukládaných dat

Požadavky na zobrazení jízdních dat v GUI:

- Zobrazení trasy jízdy na mapě
- Zobrazení jízdních údajů na grafech
- Visuální provázání jízdních údajů s mapou (zobrazení místa, kde byla hodnota zaznamenána)
- Možnost omezení zobrazovaného úseku jízdy

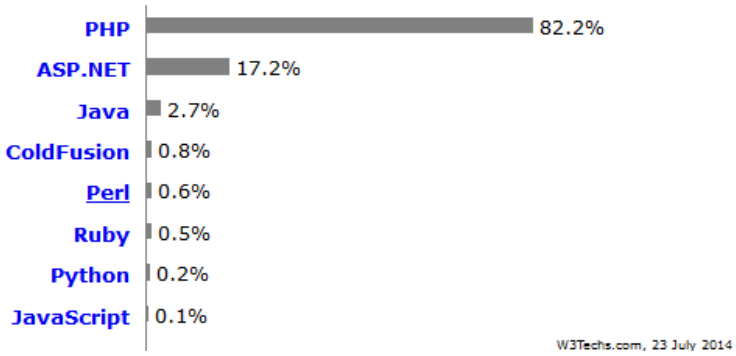
Další požadavky na GUI:

- Možnost registrace a přihlášení uživatele, validace údajů
- Zobrazení seznamu všech jízd

#### 4.1.5.2 Nefunkční požadavky

Jak již bylo řečeno (v kapitole 4.1.1), bude třeba navrhnout a implementovat novou serverovou aplikaci. To s sebou přináší i možnost vybrat serverovou platformu.

Při výběru platformy pro dlouhodobý projekt je potřeba hledět především na její dostupnost mezi poskytovateli hostingu a také rozšířenost mezi programátory, tak aby mohl být projekt bez problému nasazen a udržován.




*Obr. č. 9: Graf rozšířenost serverových programovacích jazyků, zdroj [45]*

Z grafu (Obr. č. 9), jež vychází z průběžného průzkumu webových stránek [45] vyplývá, že zdaleka nejrozšířenějším serverovým jazykem je PHP, jež stojí za víc než 80% online projektů. Dá se tak předpokládat velká uživatelská základna mezi programátory, je také k dipozici zdarma jako open-source.

Konkurenční platforma ASP.NET je spojena s vyššími náklady [46]. Programování na úrovni jazyka JAVA může být, vzhledem k jeho nižší úrovni (PHP a ASP jsou jazyky skriptovací), pro náš projekt příliš robustní.

Dle uvažovaných hledisek se jako nejvhodnější platforma jeví PHP. Vzhledem k tomu, že autor práce je s ní dobře obeznámen (a může ji vřele doporučit), bude použita pro vývoj serverové aplikace.

Požadavky na serverovou platformu

- Použití jazyka PHP verze 5.3
- Použití SQL databáze

Pozornost si zaslouží bezesporu také mapová komponenta, která bude sloužit ke zobrazení geografických dat. Nejznámější mapovou komponentou je bezesporu Google Maps API [48], avšak její podmínky neumožňují službu používat pro soukromé účely, ani účely fleet managementu [49]. V původním projektu [6] byla použita OSM [36], resp. její volně použitelná open-source mapová komponenta. Vzhledem k našemu geografickému umístění přichází do úvahy také mapové API Mapy.cz [46] firmy Seznam, které z projektu OSM vychází a které je k dispozici bez jakýchkoli omezení.

Poslední zmíně mapová komponenta se vyznačuje zřetelně menší velikostí přenesených dat a přehlednějším implementačním rozhraním nežli poskytuje OSM [36], což je pro projekt výhodné.

Požadavky na mapovou komponentu

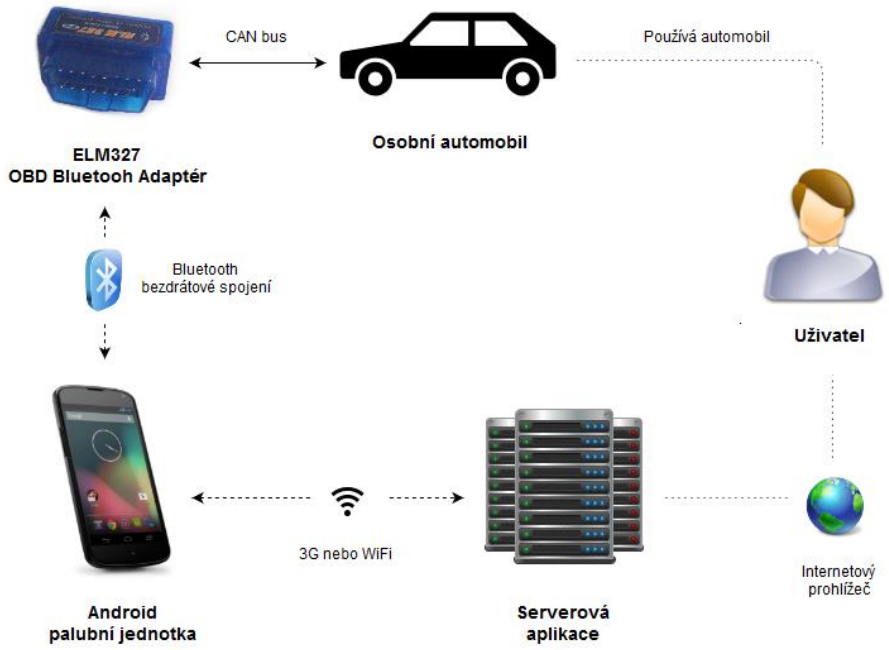
- Použití API Mapy.cz

## 4.2 Systém pro sběr jízdních dat

Palubní jednotka Android a serverová aplikace (PHP) spolu tvoří systém pro sběr dat. Jak bylo rozhodnuto (viz kapitola 4.1.2), navrhovaný systém bude sloužit pro osobní potřebu jednotlivých uživatelů, kteří jej budou používat ke sběru jízdních dat a jejich evidenci a zobrazení na webu.

### *4.2.1 Schéma systému*

Schéma primární funkce systému je zobrazeno na následujícím obrázku (Obr. č. 10).:




*Obr. č. 10: Schéma systému pro sběr jízdních dat*

Uživatel užívá vůz běžným způsobem, systém se mezi tím stará o záznam jízdních údajů. Ty potom uživatel může sledovat na webu.

Hlavní část jízdních dat obstarává OBD2 Adaptér, fyzicky připojený k vozu přes CAN sběrnici. Ten je dále bezdrátově odesílá palubní Android jednotce, kde se dočasně zaznamenávají.

Palubní jednotka data odesílá průběžně během jízdy prostřednictvím 3G internetového připojení (je-li k dispozici) nebo po skončení jízdy najednou, prostřednictvím Wi-Fi nebo 3G (je-li alespoň jedno k dispozici).

Serverová aplikace slouží k trvalému uložení jízdních záznamů. Umožňuje jejich základní organizaci a zobrazení pomocí internetového prohlížeče.

### *4.2.2 Organizace systému*

Systém sběru dat pro osobní využití předpokládá, že uživatel bude využívat jeden uživatelský účet pro jednu palubní jednotku, jež v systému představuje jeden fyzický vůz.

Palubní aplikace proto bude umožňovat přihlášení a odhlášení uživatele v případě, že by tento používal více vozů.

Serverová aplikace potom bude ukládat jízdní data pod uživatelský účet uživatele, který je v palubní aplikaci aktuálně přihlášen. Uživatel bude mít na serveru možnost zobrazení pouze záznamu jízd uložených pod svým účtem.

## 4.3 Palubní aplikace

V této sekci bude obecně popsán návrh palubní aplikace pro OS Android.

### *4.3.1 Zapojení palubní jednotky*

Ačkoli se z pohledu uživatelské funkčnosti zabýváme pouze návrhem varianty systému pro osobní použití, z hlediska pozdější rozšiřitelnosti a použitelnosti pro projekt Metrocar [3,4,7] bychom měli při návrhu počítat s ostatními variantami.

Je žádoucí, aby Android zařízení, hostující palubní aplikaci, bylo za jízdy připojeno k napájecímu adaptéru. Úkoly prováděné aplikací totiž neumožňují použití režimu spánku během jízdy, což velmi snižuje možnou výdrž zařízení na baterii. Záznam videa navíc patří z hlediska spotřeby energie k nejnáročnějším činnostem.

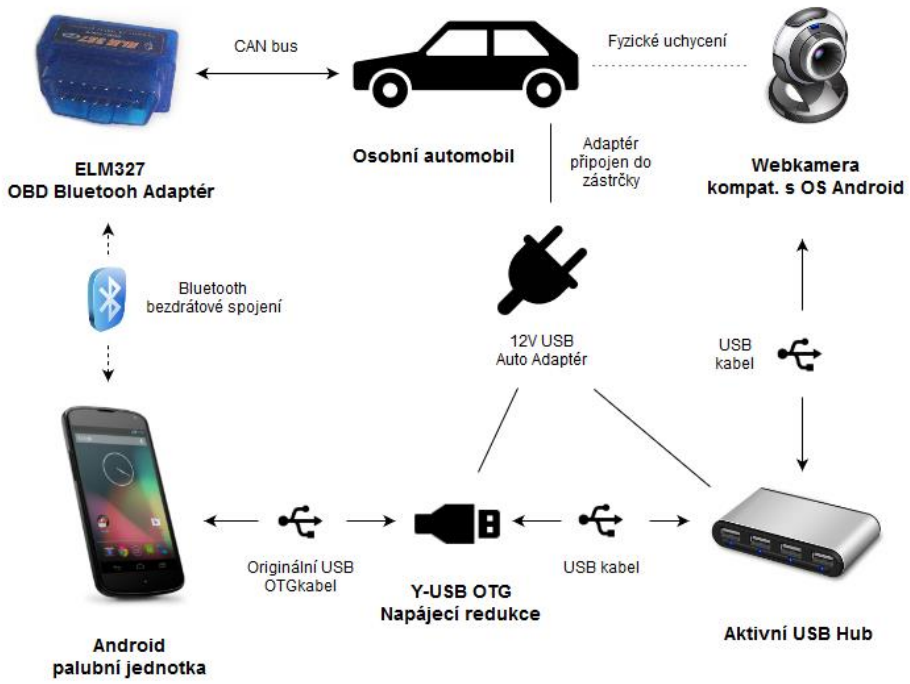
Varianta pro osobní využití předpokládá uchycení Android zařízení v odnímatelném stojanu za předním sklem. Toto by umožnilo současné využití zabudované videokamery a připojení k napájecímu konektoru.

Ve variantách pro car-sharing a fleet management se však počítá s trvalým zabudováním zařízení do vozidla. V takovém případě by bylo nutné použít pro záznam videa externí USB videokameru. Varianta car-sharing si také vyžádá použití externí USB NFC [53] čtečky (viz kapitola 6.3.2).

Naprostá většina Android zařízení však nedisponuje více než jedním USB portem, který slouží jak pro napájení, tak i pro připojení externích periférií. Zde

tedy vyvstává problém, jak zajistit aby zařízení, potažmo aplikace, mohla operovat externí periferie a zároveň bylo umožněno nezbytné napájení.

Řešením může být použití aktivního USB hubu, který umožní připojení většího množství periferií a Y-USB OTG redukce, která umožní současné připojení USB hubu a napájecího adaptéru.




*Obr. č. 11: Schéma zapojení palubní jednotky*

V praxi je takovéto zapojení za pomoci OTG považováno za problematické, přičemž mnohdy není ani možné. Vzhledem k absenci jiných možností řešení je však nutné jej implementovat a otestovat.

### 4.3.2 Návrh komponenty služeb

Aplikace bude obstarávat řadu výpočetně nebo časově náročných úloh. Tyto nemohou typicky běžet v hlavním UI vlákně aplikace [50], neboť by způsobili neresponsivitu („zamrznutí“) aplikace.

Zpoždění způsobení čekáním vlákna by také mohli způsobit nechtěné zpoždění při záznamu jízdných dat, z nichž je pro některá velmi důležitý konstantní charakter prodlevy mezi záznamy.

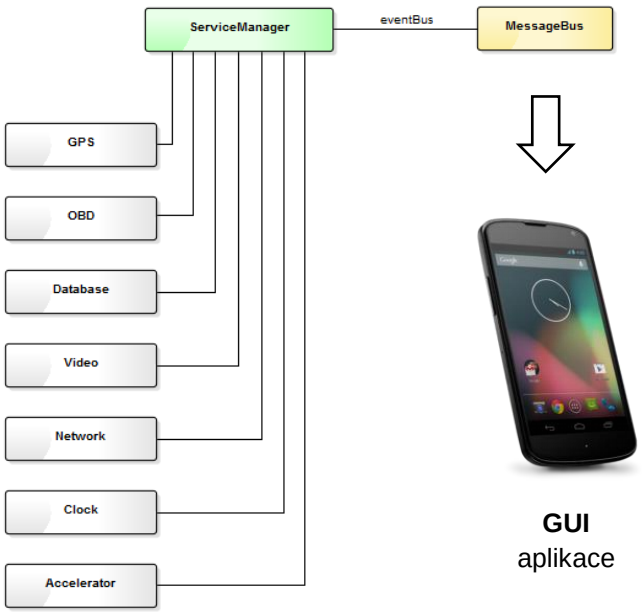
Je proto nutné navrhnout vhodnou infrastrukturu služeb tak, aby byly spouštěny v oddělených vláknech.

Vícevláknový přístup umožnuje hladší paralelní běh více operací, také ale značně komplikuje vnitřní komunikaci aplikace, neboť při mezivláknové komunikaci je vyžadována vzájemná synchronizace vláken. V prototypu [3] byla mezivláknové komunikace přítomna v mezi vláknem obstarávajícím OBD komunikaci a UI vláknem [50], realizujícím zobrazení dat v GUI. Použita byla třída AsyncTask [52], jež interně využívá předpřipravený thread pool operačního systému k vykonání akcí a komunikaci.

Tento přístup však sebou nese značný overhead spojený s potřebou vytvářet a exekvovat (vykonávat) vždy nový Runnable objekt (viz [51]). Jiný přístup k dlouhodobě běžícím úlohám také doporučuje i dokumentace ke třídě AsyncTask [50].

Z těchto důvodů bude použita architektura sběrnice zpráv (message bus, event bus), jež bude sloužit pro přenos událostí, které plně nahradí veškerou mezivláknovou komunikaci.

Situace je zachycena na následujícím diagramu (Obr. č. 12), kde všechny služby běžící ve vlastním vlákně používají jednotný event bus pro komunikaci s GUI částí aplikace.




*Obr. č. 12: Diagram systému služeb*

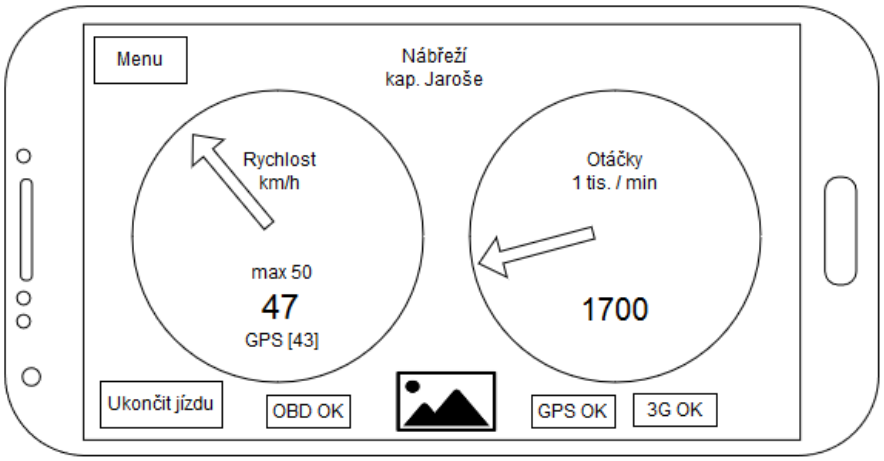
### 4.3.3 Návrh GUI

Funkce palubní jednotky, což je sběr a záznam jízdních dat, nevyžaduje sama o sobě uživatelskou interakci. Pro návrh GUI je však potřeba zohlednit sekundární činnosti, jako je přihlášení, vyvolání synchronizace, aktivace nezbytných sensorů v zařízení apod., jež se bez uživatelského potvrzení či interakce neobejdou.

Prvním místem kde uživatel přichází do styku s aplikací je tzv. „splash screen“, typicky zobrazující logo aplikace či výrobce. Slouží především k tomu, aby mohl být na pozadí inicializován systém služeb aplikace, zatímco v popředí probíhá např. jednoduchá animace.

Dalším místem bude přihlašovací formulář, kde použijeme standardní systémové komponenty (TextView, EditView a Button [56]) s výchozím vzhledem.

Pro přihlášení uživatele přejdeme na hlavní obrazovku, kde poskytneme uživateli visuální zobrazení základních měřených hodnot a informace o stavu klíčových služeb.

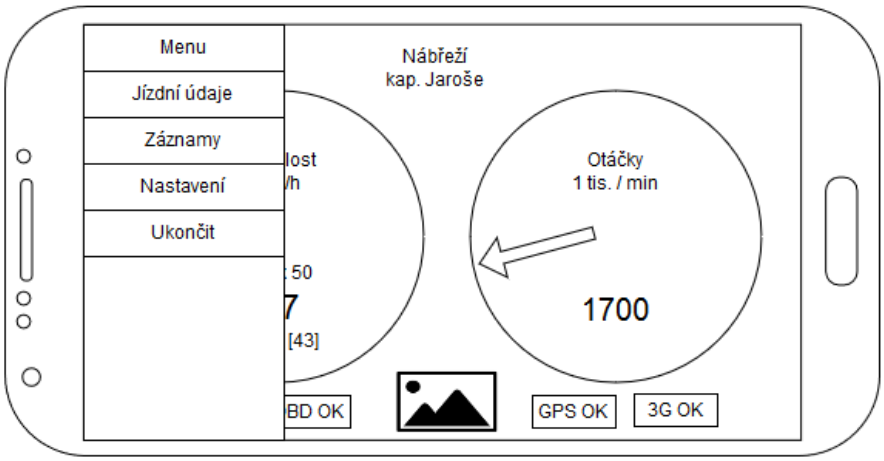



*Obr. č. 13: Návrh přístrojové desky*

Jako hlavní šablonu použijeme „Navigation drawer“ [55]. Jedná se o třídu typ uživatelské obrazovky (Activity [54]) umožňující zobrazit jiné obrazovky

(resp. fragmenty, třída Fragment [56]). Přičemž používá vysouvací menu pro přepínání mezi nimi.

Toto menu je také vhodnější oproti druhému obvyklému způsobu řešení nabídky „Options menu“ [57], neboť je visuálně větší a lze zde jen obtížně kliknout mimo.




*Obr. č. 14: Návrh přístrojové desky se zobrazeným menu*

Pro sekci Nastavení, jež bude obsahovat řadu zanořených obrazovek, použijeme automatický generované rozvržení, které se vytvoří na základě předem připraveného XML souboru. K tomuto účelu poslouží třída PreferenceActivity [56], která vytvoří celou hierarchii obrazovek nastavení. Ta se bude stávat ze zaškrtávacích polí, seznamů, jež budou plněny programově a budou využívat standardní podoby visuálních komponent.

Posledním visuálním prvkem, který je pro sekci nastavení nutné navrhnout je editační dialogů, které bude sloužit pro vytvoření nebo úpravu parametrů OBD PID kódů (Obr. č. 15).

Bude se rovněž skládat ze standardních komponent, avšak jejich rozvržení bude třeba definovat v odděleném XML souboru. U rozvržení je také třeba dbát na to, aby byl na obrazovce prostor zobrazení softwarové klávesnice, nezbytné pro zadávání údajů a aby takto nezakrývala důležitá pole nebo tlačítka.

*Obr. č. 15: Návrh editačního dialogu PID*

### 4.3.4 Vývojové prostředí

Vzhledem k velkým odlišnostem prototypu palubní aplikace [3] a její nově navržené podoby (viz výše) v ohledech grafického rozhraní i systému služeb, je z hlediska efektivity výhodnější vytvoření nového aplikace. Počet zachovaný částí původního kódu by byl pravděpodobně velmi malý nebo nulový.

Tato skutečnost dovoluje také možnost změny vývojové prostředí. S ohledem na oficiální doporučení firmy Google (viz 58), bylo zvoleno IDE Android Studio, jako nová vývojová platforma.

Aplikace bude vytvořena jako nový projekt v IDE, nebude tedy zapotřebí migrace projektu.

## 4.4 Serverová aplikace

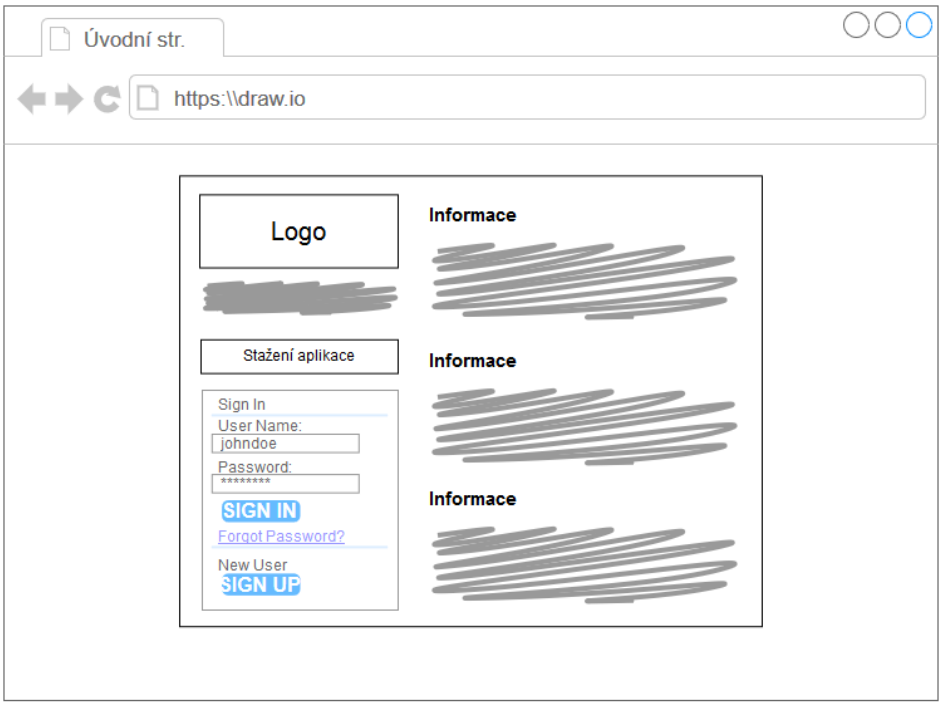
V této části bude obecně navržena podoba serverové PHP aplikace a jejího API.

### 4.4.1 Uživatelské rozhraní

GUI serverové aplikace bude rozděleno do dvou částí. První bude přístupná bez přihlášení a bude se skládat pouze z úvodní stránky. Bude umožňovat uživatelů získat základní informace o projektu.

Noví uživatelé zde budou mít možnost registrace nové uživatelského účtu. Ti stávající se budou moci přihlásit nebo přímo instalovat palubní aplikaci pro osobní použití do svého Android zařízení.

Návrh úvodní stránky bez registračního formuláře (Obr. č. 16).:

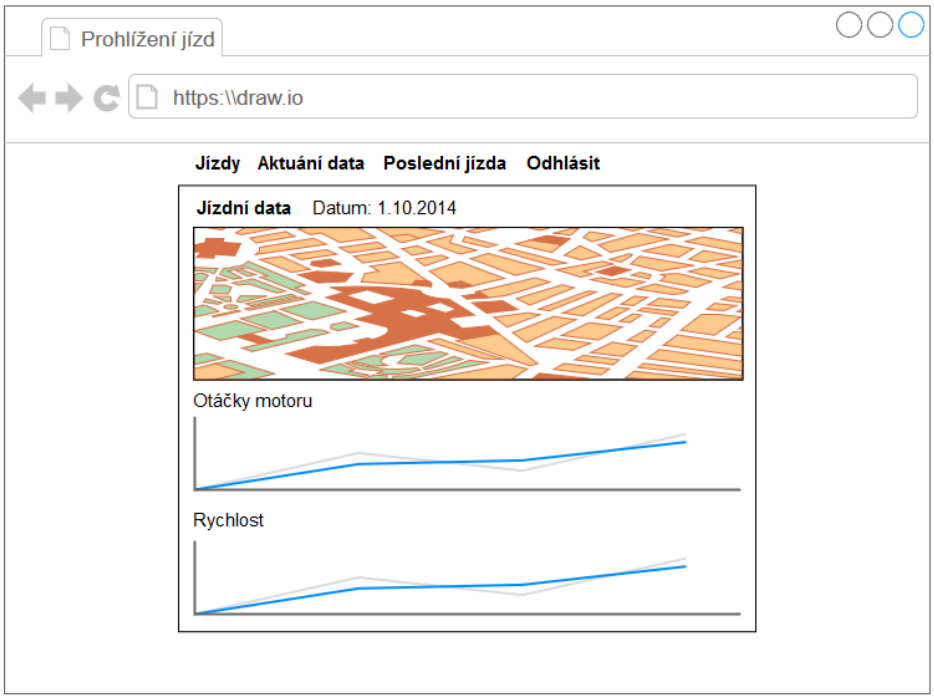



*Obr. č. 16: Návrh serverové aplikace – úvodní stránka*

Pro registrované uživatele bude k dispozici druhá část aplikace, která bude přístupná po přihlášení.

Zde bude zobrazen seznam jízd, provedený pod přihlášeným uživatelským účtem. Kliknutí na jednotlivé položky potom zobrazí data z příslušné jízdy na mapě a grafech (viz Obr. č. 17).

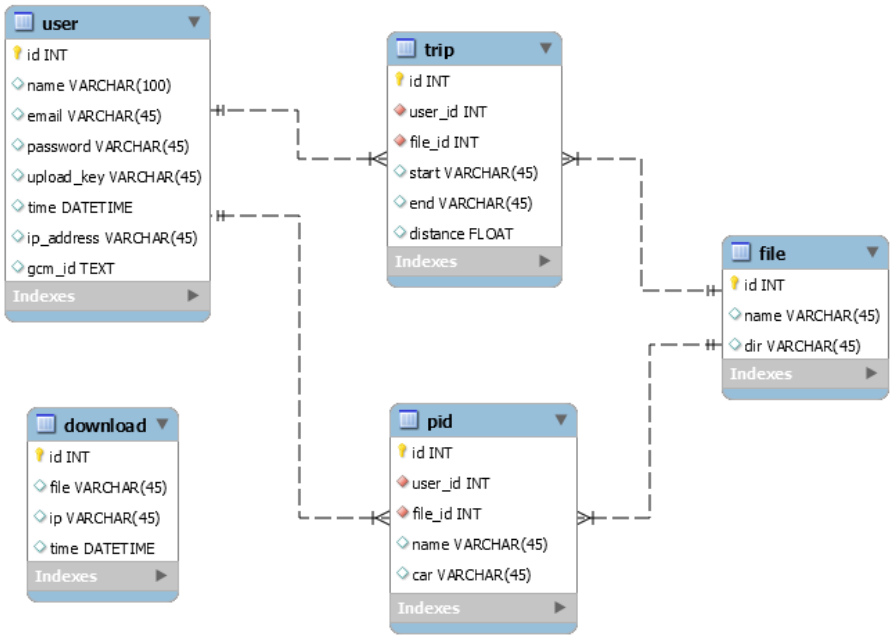
K dispozici také sekce pro zobrazení aktuálních jízdních dat, kterou bude uživatel moci vzdáleně sledovat data z vozu. Tato funkce se může hodit např. v případě, že vozidlo bylo zapůjčeno rodinnému příslušníkovy apod. Funkce také především pro demonstraci, neboť praktičtější by byla především ve verzích car-sharing a fleet management.




*Obr. č. 17: Návrh serverové aplikace – prohlížení jízd*

### 4.4.2 Ukládání dat

V serverové aplikaci se předpokládá použití MySQL [59] databáze. Ta bude sloužit k uložení uživatelských dat souvisejících s uživatelskými účty, nestandartními PID [27] kódy a jednotlivými jízdami.




*Obr. č. 18: Schéma databáze serverové aplikace*

Do databáze naopak nebudou ukládány přímo záznamy jednotlivých jízdních údajů. Toto rozhodnutí bylo učiněno s ohledem na skutečně velký počet zaznamenávaných jízdních údajů (viz 6.3.3), který může snadno dosáhnout velikost několika MB za jednu jízdu.

Celkové údaje z jízdy budou uloženy v souborovém systému serveru (jako běžné soubory) s příponou „.json“, což naznačuje, že použijeme JSON [60] syntaxi jako datový formát.

Stejná situace bude také u aktuálních dat z probíhající jízdy, jež budou uloženy také ve formě souboru. Tento však bude s každými příchozími údaji přepisován, tak aby obsahoval vždy pouze ty nejaktuálnější.

Ve formě JSON souborů budou ukládány také uživatelské definice nestandartních PID kódů [27].

Členění souborového systému do složek dle uživatelů a typu dat je specifikováno v implementační iteraci (v kapitole 6.1.2).

### *4.4.3 API*

Pro komunikaci mezi palubní jednotkou a serverovou částí musí existovat společné rozhraní. Toto rozhraní bude nabízet serverová aplikace a bude dostupné na předem stanované URL serveru, které bude přístupné bez přihlášení.

Palubní aplikace se zde bude autorizovat ID uživatele a klíčem („upload_key“, viz Obr. č. 18), jež budou ověřeny oproti databázi.

Typy API požadavků:

- Ověření uživatele (získání klíče pro palubní aplikaci)
- Stažení standardních PID kódů a vzorců
- Odeslání aktuálních jízdních dat
- Odeslání souboru s celkovým záznamem jízdy

Dotazy vždy budou typu HTTP POST [61] s daty kódovanými v notací JSON. Konkrétní podoba dotazů na API bude specifikována během implementační iterace.

# 5 Vývoj palubní jednotky

Vývoj software pro palubní jednotku je třetí iterací iterativní přístupu. Řešena bude především implementace a testování aplikace.

## 5.1 Vývoj GUI

Uživatelské rozhraní se v prostředí systému android obvykle implementuje jako rozšíření třídy Activity [54] (budeme označovat jako „aktivitu“). Ta obvykle představuje jednu konkrétní obrazovku týkající se určité činnosti v aplikaci.

Visuální prvky jsou do aktivity přidávány buďto z předem definovaného XML souboru, jež obsahuje definic layoutu [62] nebo přímo v kódu rozšiřující třídy programově, instanciací tříd odpovídajících visuálním komponentám v XML (např. TextView pro text, Button pro tlačítko apod.).

### *5.1.1 UI Manager a splash screen*

Jako vstupní bod do aplikace použijeme aktivitu SplashScreen. Ta bude sloužit pro zobrazení krátké animace loga projektu, zatímco se v mezičase bude inicializovat systém služeb.

K tomu aby byla aktivita použita jako vstupní interpretem operačního systému, je třeba nastavit v manifestu (XML soubor AndroidManifest [63]) příslušný „intent-filter“ [64]. Ten slouží operačnímu systému k identifikaci

aktivity, jež má vyvolat pro externě vyvolanou akci týkající se aplikace.
V případě spuštění je to akce (resp. objekt třídy Intent [56]) MAIN z kategorie LAUNCHER [56], viz Obr. č. 19).

```
<application
   . . .
     <activity
       android:name=".ui.activity.SplashActivity"
       android:label="@string/app_name"
       android:noHistory="true" >
       <intent-filter>
         <action
           android:name="android.intent.action.MAIN" />
         <category
           android:name="android.intent.category.LAUNCHER" />
       </intent-filter>
     </activitv>
```

*Obr. č. 19: Definice vstupního bodu aplikace*

Vstupní aktivitu použijeme k inicializaci třídy UIManager, což je naše třída obstarávající záležitosti spojené s fungování GUI napříč všemi aktivitami v aplikaci. Zde je také na pozadí inicializován objekt systému služeb.

Mezi funkce UIManageru patří například nastavení režimu celé obrazovky nebo nastavení parametrů bránicí zařízení ve vypnutí obrazovky a přechod do režimu spánku (což by přerušilo záznam jízdních dat).

Funkcí UIManageru je centrální evidence instancí všech aktivit a fragmentů, což bude dále použito při přepnutí obrazovek při výběru položek vysouvacího menu.

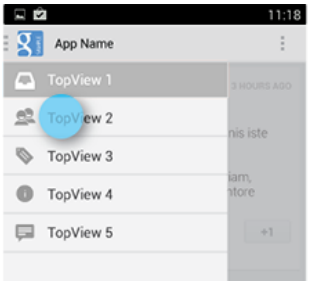
Samotnou visuální podobu animace (na Obr. č. 20) bude obstarávat SplashView, rozšiřující super-třídu View všech visuálních objektů umístěných v aktivitě či fragmentu [56].

*Obr. č. 20: Úvodní obrazovka palubní aplikace*

### 5.1.2 Menu a navigace

Hlavní aktivitou aplikace je potom MenuActivity, jež vychází z návrhové šablony „Navigation drawer“ [55] (viz Obr. č. 21). Ta používá vysouvací menu pro zobrazení položek navigace, jež lze vyvolat buďto kliknutím na ikonu aplikace v „akční liště“ (ActionBar [56], lišta s ikonou, názvem aplikace a nabídkou „Options menu“ [57]) nebo tažení ze strany dotykového displeje zařízení.

Tento způsob navigace budeme používat jako jediný, abychom mohli zcela odstranit akční lištu, zabírající místo na displeji (což činí problém zejména při horizontální orientaci, kdy akční lišta zabírá větší percentuálně podíl obrazovky nežli při orientaci vertikální).




*Obr. č. 21: Vysouvací menu šablony Navigation Drawer, zdroj [55]*

Tento způsob navigace budeme používat jako jediný, abychom mohli zcela odstranit akční lištu, zabírající místo na displeji (což činí problém zejména při horizontální orientaci, kdy akční lišta zabírá větší percentuálně podíl obrazovky nežli při orientaci vertikální).

Výběrem položky z menu bude docházet k přepnutí vizuálních fragmentů, jež byly předem připraveny během inicializace aplikace. O vyhodnocení akce prokliku se stará UIManager, jež je volán označení položky a na základě jejího indexu zvolí vhodnou akci. Ten může kromě přepnutí fragmentu vyvolat po kliku na položku i jiné akce (např. ukončení aplikace při kliknutí na „Ukončit“ apod.).

### 5.1.3 Nastavení a dialogy

Důležitou součástí GUI jsou dialogy. V aplikaci jsou použity na místech, kdy stav systémových komponent v zařízení neodpovídá požadovanému stavu. K tomu může dojít např. pokud se uživatel pokouší o přihlášení k aplikaci, bez aktivního internetové připojení nebo pokud má být pro jízdu použit záznam polohy z GPS, avšak GPS subsystém je v zařízení deaktivovaný.

*Obr. č. 22: Uživatelské dialogy*

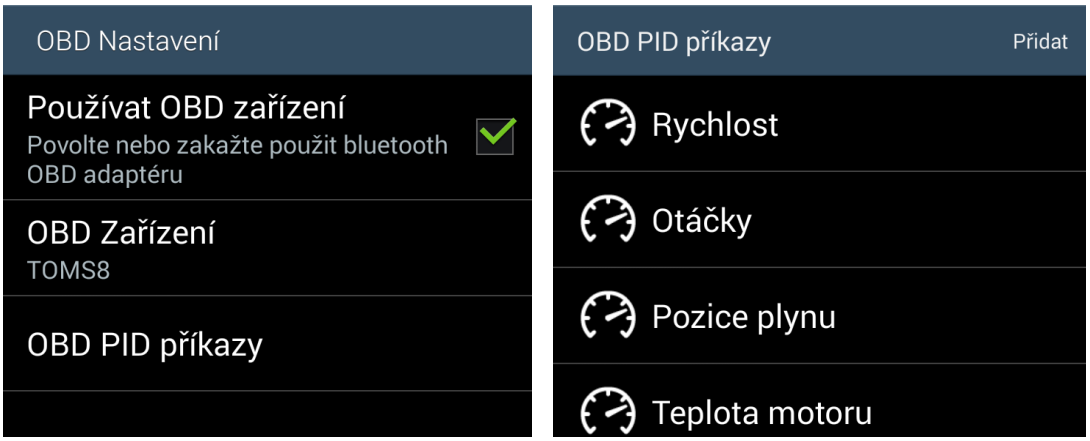
Dialogy jsou implementovány za použití pomocné třídy DialogBuilder [56], jež používá standartní systémové komponenty k vytvoření visuálních prvků.

Většina dialogů v aplikaci však nahrazuje standartní textovou komponentu, sloužící ke zobrazení textu výzvy, vlastním TextView [56], které je použito pro zobrazení formátovaného textu. Tento slouží rozlišení mezi hlavní výzvou a doplňujícími informacemi (viz Obr. č. 22).

Obrazovky nastavení jsou implementovány jako samostatná aktivita SettingsActivity jež rozšiřuje super-třídu PreferenceActivity [56], jež je pro tyto

účely určena. Používá předem strukturu voleb, předem definovanou v XML souboru („/res/xml/my_settings.xml“) . Tato je následně třídou SettingsActivity upravena o pokročilejší možnosti, jako např. možnost zakázání či povolení funkčnosti celé sekce nastavení nebo okamžité přidání nové položky do nabídky.

Obrazovky nastavení jsou zachyceny na následujícím obrázku (Obr. č. 23).:




*Obr. č. 23: Obrazovky nastavení*

Obsah pro obrazovky nastavení je získáván prostřednictvím databázové služby (viz 5.2.7). Po změně nastavení dochází k v databázi, případně v místním uložišti aplikace (dle typu nastavení) ke změně hodnoty.

Nově změněná hodnota je promítnuta do fungování aplikace jakmile uživatel opustí nastavení a přejde zpět na hlavní obrazovku (MenuActivity). O realizace změny hodnoty a celkově o synchronizace mezi nastavením a reálným stavem komponent aplikace se stará UserController (viz 5.3.1).

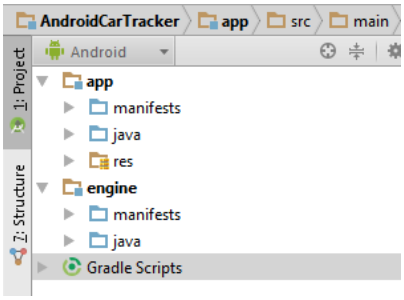
## 5.2 Systém služeb

Systém služeb pro palubní aplikaci je navržen jako knihovní aplikace universálně použitelná napříč všemi variantami produktu (4.1.1).

### *5.2.1 Struktura projektu v IDE*

V rámci IDE je systém služeb obsažen v modulu „engine“ v samostatném projektu s názvem „AndroidCarTrackerEngine“. Jedná se však o knihovní aplikaci, která není sama o sobě spustitelná, slouží pouze pro využití jiným projektem.

Hlavní IDE projekt aplikace nese název „AndroidCarTracker“, zde je přítomen modul „app“, jež obsahuje GUI část (posanou v předchozí kapitole 5.1). Modul služeb je do toho projektu naimportován (viz Obr. č. 24).:




*Obr. č. 24: Struktura projektu v IDE*

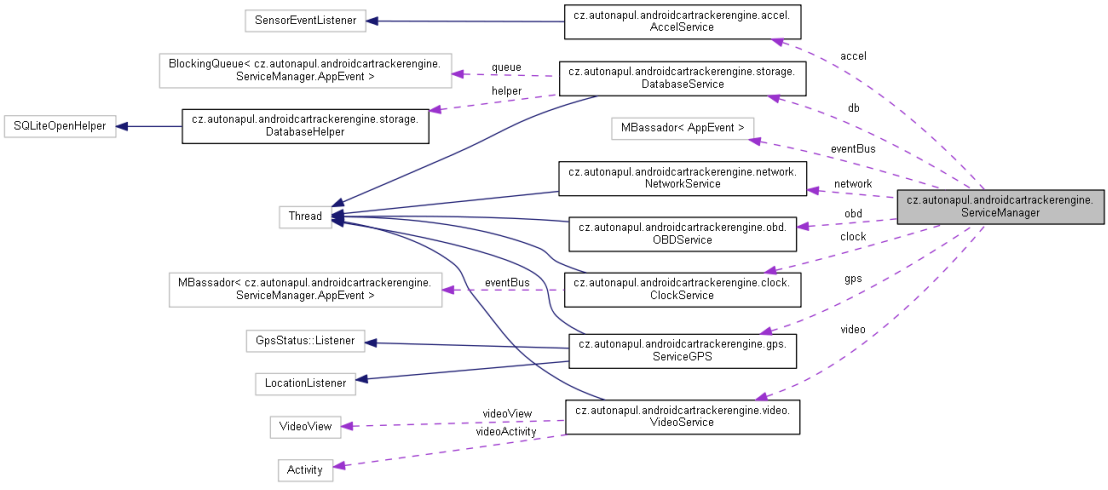
Jakkoli se může toto členění zdát banální, jeho provedení vyžadovalo poměrně velké úsili při konfiguraci buildovacího systému Gradle [65].

### 5.2.2 Architektura systému

Systém služeb (na Obr. č. 25) nabízí následující služby:

- OBDService – služba pro získávání dat z OBD adaptéru
- GPS Service – služba pro příjem GPS dat
- StorageService – služba pro záznam dat do databáze
- NetworkService – služba pro internetovou komunikaci a konektivitu
- VideoService – služba pro záznam video
- AccelService – služba pro získávání dat z akcelerátoru
- ClockService – služba generující časové signály

Vstupní třídou k systému služeb je ServiceManager, dle návrhového vzoru singleton [66], umožnující existenci pouze jedné instance. To odpovídá scénaři




*Obr. č. 25: Schéma systému služeb palubní aplikace.*

použití, kdy požadujeme existenci pouze jedné instance každé služby, neboť existence dalších instací kterékoli služby by způsobila více násobné odeslání generovaných událostí na sběrnici údálostí (viz dále v kapitole 5.2.3).

Většina služeb je implementována jako samostatné vlákno (rozšířením JAVA [75] třídy Thread [76]). Služby existují jako jediná instance ve třídě ServiceManager, jež je sdružuje. Zde jsou vytvořeny, inicializovány, ale nemusí být spouštěny (metoda „start“), neboť spuštění služeb mohou iniciovat také controllery.

Životní cyklus služeb končí vyvolání metody „exitApp“, jež nastaví příznak ukončení všem službám s vlastním vláknem. Poté dojde ke sjednocení vláken (metoda „join“ u super-třídy Thread) s timeoutem 500ms.

Objekty služeb takto zůstávají přístupné i po ukončení jejich vnitřních cyklů, což zamezí chybám typu („NullPointerException“ [78]), v případě, že by se objekty z GUI části aplikace pokoušeli volat některou ze služeb po té, co již bylo vyvoláno ukončení aplikace.

### 5.2.3 Sběrnice událostí

GUI část aplikace může volat dostupné služby přímo, využitím statické třídy ServiceManager (resp. metody „getInstance“). Služby ale naopak nemohou přímo volat žádné metody či objekty z GUI části, neboť o nich nemají žádné povědomí (jsou od nich odstíněny) na úrovni kódu (této technice se také říká „loose coupling“ [9]).

GUI část aplikace však nezbytně potřebuje odezvu od využívaných služeb, ať již pro aktualizaci zobrazených dat nebo např. pro zjištění stavu komponent zařízení a mnoho dalších využití.

V systému služeb je proto implementována sběrnice událostí (event bus, nebo také message bus) MBassador [67], jež je k volně dispozici pod MIT licencí [68]. Architektura tohoto typu sběrnice umožňuje třídám v GUI části aplikace, aby se registrovali pro příjem požadovaného typu zpráv pomocí anotovaných metod [69]. Příklad použití sběrnice je na dalším obrázku (Obr. č. 26).:

```
public MyObject(){

   // registrace k odběrů událostí
   ServiceManager.getInstance().eventBus.subscribe(this);

}

@Handler
public void handleOBDPID(final OBDService.OBDEventPID evt){

   if( evt.getType() == TYPE_SPEED ){
      AppLog.i( "Aktuální rychlost: " + evt.getValue() );
   }
}
```

*Obr. č. 26: Použití sběrnice událostí*

Sběrnice tedy umožní systému služeb nepřímou komunikaci s GUI částí aplikace, aniž by byla vyžadována apriori znalost objektů a tříd GUI části (loose coupling). Tento přístup nám zaručí nejen přehlednější kód i architekturu [9], ale

umožní také větší flexibilitu při implementaci aplikace pro další varianty produktu (viz 4.1.2) v budoucnu.

### 5.2.4 Služba OBD

Služba OBD vychází na nejnižší úrovni ze své předchůdkyně v prototypu aplikace [3]. Používá běžné metody pro získání zařízení spárovaných s Android zařízením [70], ověření jejich stavu a připojení.

Služba také umožnuje aktivovat Bluetooth subsystém v zařízení, aniž by vyžadovala další uživatelovu interakci. V případě, že ale došlo k přerušení párování s Bluetooth zařízením, je vyžadována interakce ve formě potvrzení PIN kódu zařízení (u testovaného BT adaptéru [28] byl kód „1234“).

Po získání párovaných BT zařízení bezprostředně po inicializaci služba vyčkává, dokud není nastartována kontolerem. Po startu vlákna je navázáno spojení s BT OBD adaptérem na nejnižší úrovni, a dojde vytvoření příchozího a odchozího datového proudu.

Následně vlákno vstupuje do hlavní smyčky, kde je v každém cyklu nejprve ověřen stav spojení (případně provedeno opětovné připojení). Tímto služba řeší situace s výpadně BT spojení, k nimž může během jízdy výjimečně dojít.

Druhou částí hlavní smyčky je postupné odesílání seznam PID kódů (proměnná „queue“) a jejich interpretace (viz následující kapitola 5.2.5). Pokud tato proběhne v pořádku, jsou opět odeslány při další cyklu, atd. Pokud však dojde u všech dotazů k chybě, je nastaven příznak pro provedení opětovného

připojení („reconnectNeeded“, viz. **Chyba! Nenalezen zdroj odkazů.**), neboť služba předpokládá, že spojení bylo přerušeno.

V případě ukončení jízdy nebo aplikace je nastaven příznak pro ukončení threadu („threadRun“), jež zamezí hlavnímu cyklu v provední další iterace. Tento způsob ukončení je shodný u všech služeb s vlastním vláknem.

### *5.2.5 Interpretace PID kódů*

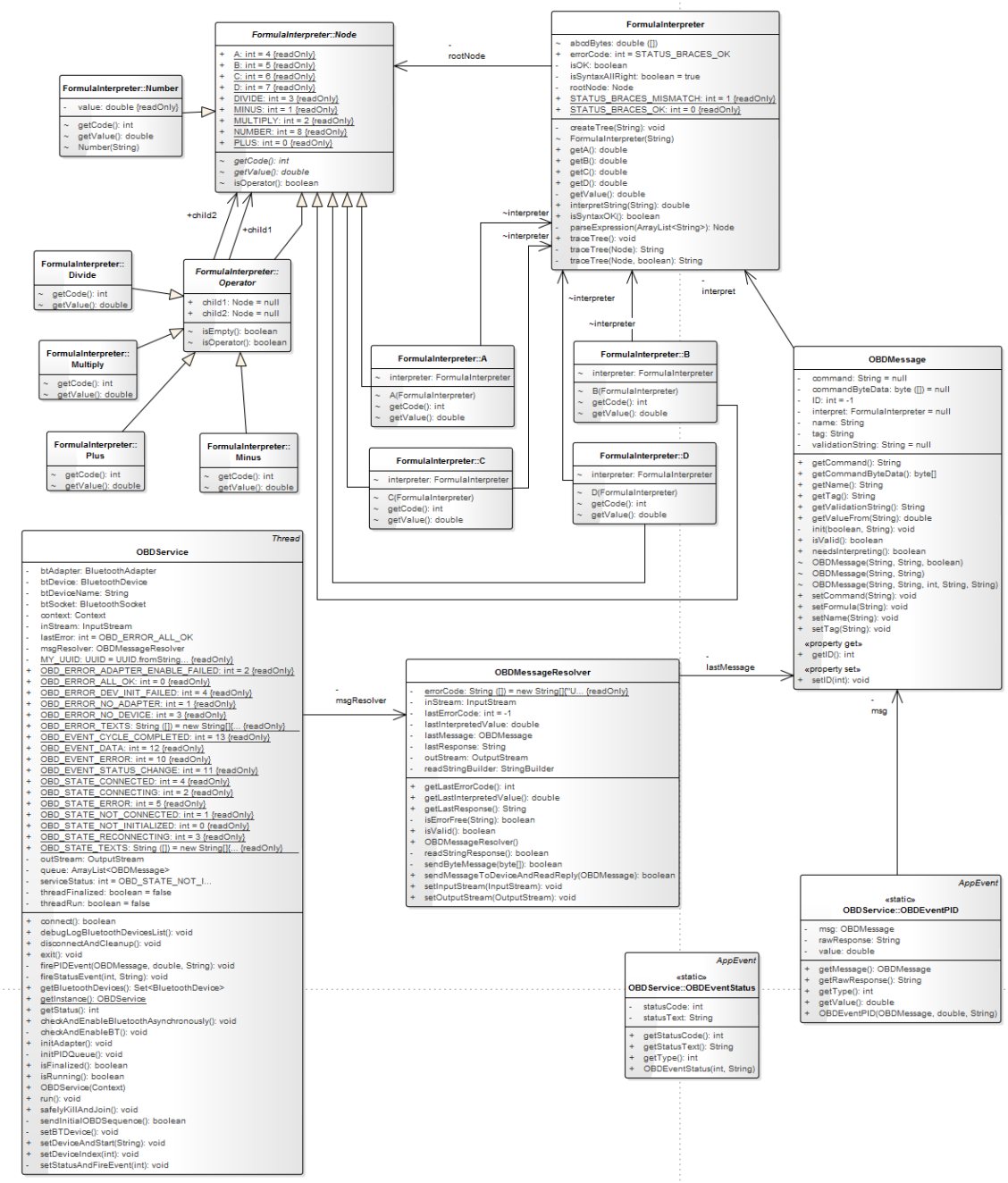
Tato část aplikace je zásadní pro její kompatibilitu s nestandartními PID kódy [27] a různými vozy obecně.

Interpret je implementován třídou („FormulaInterpreter“ viz. Obr. č. 27), která umoňuje interpretaci příchozí bytové odpovědi na číselnou hodnotu, a to dle textově zadaného vzorce. Jedná se tedy vlastně o dosazení proměnných do rovnice a výpočet výsledku.

Dle použití (zachyceno na diagramu Obr. č. 27) disponuje služba OBD jedním objektem typu MessageResolver, jež slouží pro odeslání a přijetí bytové odpovědi z datového proudu OBD BT adaptéru (resp. z ECU vozu).

Tento postupně zpracovává zprávy typu OBDMessage (existující téže v jedné instanci), přičemž každá OBDMessage má právě jednu stálou instanci třídy FormulaInterpreter, jež si vytvořila při své inicializaci.

FormulaInterpreter potom zpracovává přijatou bytovou odpověď, jež MessageResolver delegoval na přísloušnou OBDMessage. Používá přitom stromovou strukturu elementárních tříd, jež si sestavil při své inicializaci.




*Obr. č. 27: Diagram služby a interpretu OBD zpráv*

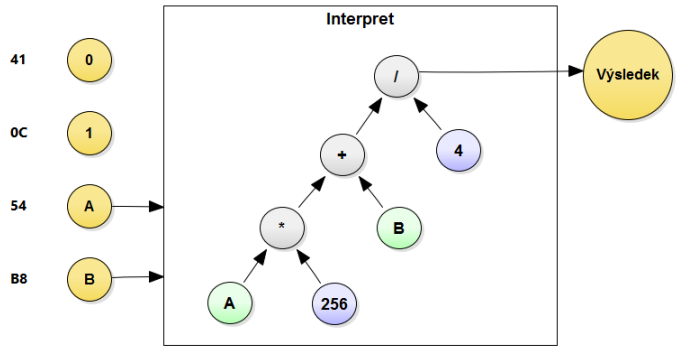
*(pro větší názornost designu klíčové komponenty přiloženo k textu)*

Vzhledem k faktu, stromová struktura interpretu zůstává po počáteční inicializaci beze změny, je takové řešení velmi výpočetně efektivní.

Při interpretaci nově přijaté zprávy jsou pouze přepsány proměnné odpovídající listům stromu a rekursivně přečten výsledek. Na následujícím diagramu (Obr. č. 28) je zachycena situace kdy máme v interpret incializovaný pro vzorec „((A*256)+B)/4“, což je vzorec pro výpočet otáček motoru.

Mějme příchozí zprávu ve tvaru „41 0C 54 B8“, což jsou 4 číslice v hexadecimálním zápisu (odpovídá 4 bytům dat). První byte označuje typ PID dotazu, druhý potom konkrétní dotaz, tyto dva tedy nejsou při výpočtu uvažovány. Další dva byty potom představují datové hodnoty, třetí byte se označuje jako „A“, čtvrtý jako „B“ (interpret podporuje max. 4 bytová data, tedy ještě „C“ a „D“ pro náročnější dotazy).

Přečteme decimální hodnoty bytů A (84) a B (136). Tyto vložíme do stromu a načteme výsledek z nodu „/“. Ten mezi sebou dělí hodnotu z nodu „+“ a „4“ atd. pro celý strom, až dostaneme hodnotu 5410.




*Obr. č. 28: Diagram interpretace PID kódu*

Tento způsob výpočtu je nejen universální pro všechny PID kódy, ale také velmi rychlý. Interpretace trvá průměrně 0.4 ms, což je jen zlomek doby potřebné k načtení OBD dotazu z vozu (což je dle pozorování průměrně zhruba 200 ms pro jednobytovou a 300 ms pro dvou bytovou odpověď, dle [3] 250 ms a 340 ms).

### 5.2.6 Síťová služba

Síťová služba umožňuje síťovou komunikaci a zasílání souborl přes internet. Je využívána přímo prostřednictvím metod „sendRequest“ a „sendFileToServer“. Tyto vyvolají zařazení dotazu do fronty.

Fronta dotazů je vyřizována v hlavním cyklu vlákna, zde dochází k vlastní přípravě těla zprávy včetně načtení souboru nebo JSON kódování dotazu.

Připravený dotaz je poté odeslán na požadovanou URL adresu, a je načtena odpověď. Ta potom vyvolá vytvoření události na sběrnici událostí, jež obsahuje výsledek dotazu.

GUI část potom může snadno naslouchat příslušnému typu události („NetworkRequestEvent“ pro JSON dotaz a „NetworkFileUploadEvent“ pro upload souboru. K identifikaci konkrétního dotazu mezi více objekty používající službu slouží textový parametr „id“ (metoda „getID“).

### 5.2.7 Služba pro ukládání dat

Tato služba je implementována ve třídě „StorageService“. Její podoba je stejná jako u síťové služby. Využívá synchronizovaný zásobník do něhož se přidávají požadavky na uložení dat.

V hlavním cyklu vlákna jsou tyto požadavky potom vykonávány asynchronně a je odeslána událost o změne počtu záznamů v DB.

Tento přístup byl zvolen s ohledem mírné pozdění, které bylo pozorováno při zcela synchronním zápisu. Ačkoli se jednalo o jednotky (až max nízké desítky) ms, tak zhledem množství a frekvenci záznamu dat (desítky za vteřinu) bylo již celkové spoždění výrazné a dobře pozorovatelné (pomalá odezva GUI).

Naopak při asynchronním přístupu nečiní zpoždění žádný problém, neboť práce s databázi probíhá v jiném vlákně.

## 5.3 Využití služeb v GUI

Systém služeb nabízí poměrně bohatou funkcionalitu. S minimální konfigurací či interakcí z GUI části je schopen samostatně zaznamenávat a ukládat data z OBD, GPS i akcelerometru.

Vzhledem k osobní povaze využití aplikace je však nutné uvažovat určitou podmíněnou interakci mezi uživatelem a systémem služeb. Je např. žádoucí, aby uživatel mohl deaktivovat službu OBD pokud nevlastní vyžadovaný BT adaptér. Uživatelé vyžadující soukromí by naopak mohli požadovat deaktivaci GPS apod.

Z těchto důvodů bylo nutné implementovat mechanismus nastavení.

### 5.3.1 Nastavení a kontrolery

Uživatelská nastavení jsou uložena výhradně v persistentní paměti aplikace [79]. Ta je pro větší pohodlí programátora přístupna přes metody („set“ a „get“) statické třídy „DB“.

O promítnutí nastavení do skutečného stavu služeb se starají kontrolery, především třída „UserController“, jež obstarává veškerá nastavení týkající se uživatele. A dále „TripController“, sledující stav aktuální jízdy.

Tento návrh vychází z architektury MVC [80], avšak v našem pojetí se jedná o komponentu komunikující zároveň se službami a zároveň mající možnost přímé komunikace s uživatelem prostřednictvím aktuálně zobrazené aktivity.

Této vlastnosti je využito např. v situaci, kdy má být dle uživatelského nastavení aktivní v zařízení subsystém GPS, ale ve skutečnosti je deaktivovaný. Kontroler má možnost v takovém případě nabídnout uživateli dialog, ve kterém může GPS zapnout.

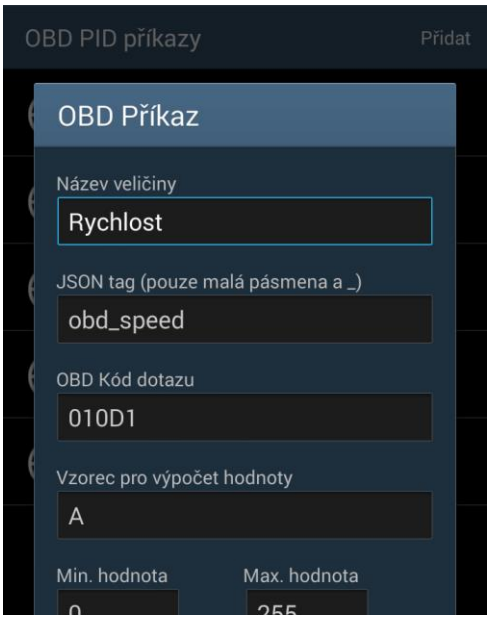
V obráceném směru se také kontroler stará o to, aby se projevila uživatelská nastavení. Po startu aplikace si tedy např. „UserController“ načte nastavení týkající se OBD a pokud má být služba vypnuta, nepřikročí vůbec k jejímu spuštění.

### 5.3.2 Uživatelské PID kódy

Komponenta umožňující zobrazení a editaci seznamu PID kódů je další ukázkou spolupráce mezi GUI a službami. Pro zobrazení seznamu si aktivita nastavení vyžádá od služby pro ukládání dat (StorageService), seznam PID kódů.

Na stejné obrazovce lze kódy také přidávat, což databázová služba reflektuje vytvořením nového záznamu v databázi (kopie PID kódu pro získání rychlosti).

Při editaci nejprve proběhne načtení dat z DB služby dle zvoleného záznamu. Po ukončení editace pak proběhne podle volby uživatele uložení nebo stornování nových dat nebo vymazání celého PID kódu. Uložení nebo vymazání má opět na starosti databázová služba.



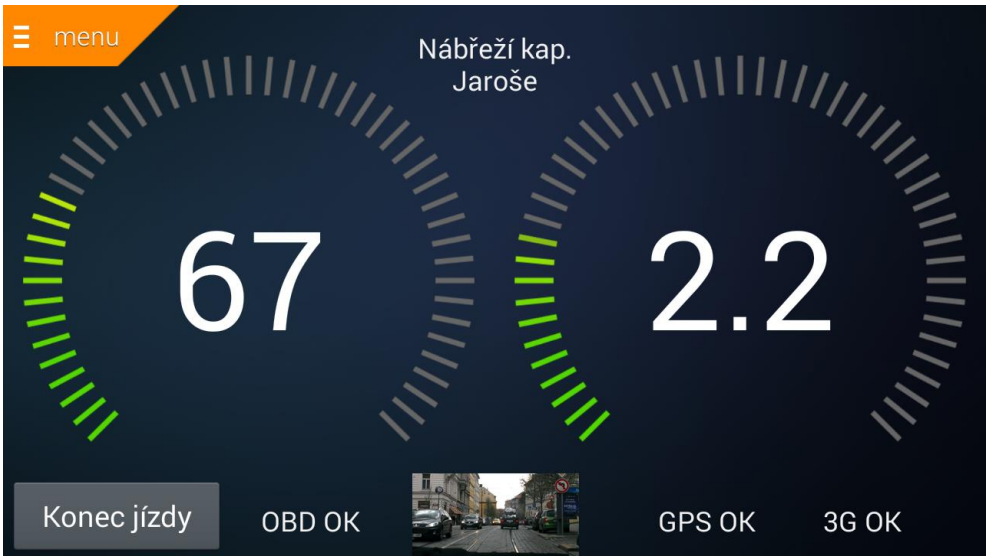

*Obr. č. 29: Obrazovka editace PID kódu*

### 5.3.3 Přístrojová obrazovka

Jedná se o hlavní jízdní obrazovku aplikace. Je navržena pro trvalé zobrazení během jízdy, pozadí je provedeno ve tmavých barvách, tak aby zbytečně nerušilo uživatele jasem displeje.

Visuálně zde dominují dva animované ukazatele. Ukazatel vlevo zobrazuje rychlost, ukazatel vpravo potom otáčky motoru získané z OBD.

Hlavním účelem obrazovky je ale poskytnout uživateli přehled o stavu sensorů, aby si mohl snadno potvrdit, že se jízda v pořádku zaznamenává. Textově jsou znázorněny stavy OBD a GPS služby a stav internetového připojení. Visuálně si lze také potvrdit záběr videokamery zařízení, např. pro korekci umístění přístroje ve voze nebo pro ověření správné funkce kamery.




*Obr. č. 30: Přístrojová obrazovka*

Obrazovka je implementována třídou „DashboardFragment“, což je fragment zobrazený v hlavní aktivitě. Ke změně údajů o stavu služeb, stejně jako ke změně hodnot číselných parametrů dochází po obdržení událostí ze sběrnice.

U náhledu videa přenos dat řídí přímo framework operačního systému, již od inicializace, není tedy třeba přenášet obrazová data po sběrnici.

## 5.4 Testování

### *5.4.1 Android zařízení*

Palubní aplikace byla testována na těchto zařízeních:

- (Z1) HTC One S, Android 4.1.1
- (Z2) Prestigio 5570C, Android 4.1.1
- (Z3) Samsung S III, Android 4.3
- (Z4) Samsung S III Neo, Android 4.4.2
- (Z5) Google Nexus 7 (výr. Asus), Android 4.4.4

### *5.4.2 Vozy*

Palubní aplikace byla testována v těchto vozech (na zařízeních):

- (V1) Alfa Romeo 156, diesel, 103kW, r.v. 2004 (Z1 – Z5)
- (V2) Audi A4, diesel, 105kW, r.v. 2012 (Z1 – Z5)
- (V3) Škoda Fabia, diesel, 47kW, r.v. 2001 (Z1, Z4)
- (V4) Hyundai i20, diesel, 55kW, r.v. 2012 (Z1, Z4)

### 5.4.3 Testovací případy

Uživatelské testování aplikace probíhalo dle předem připravených scénářů, vždy oproti OBD simulátoru [81] (více zde [3]) nebo v reálném voze pokud si to scénář vyžadoval. Při testování mimo dostupnost GPS signálu byla použita simulace polohy v zařízení.

Pro běžný vývoj byla aplikace testována se zařízeními (Z1 a Z4) v simulátoru. Větší modifikace potom pro všechna zařízení s vozy (V1 a V2). Závěrečná verze potom pro všechny vozy a zařízení.

#### 5.4.3.1 Inicializace testování

Před začátkem testování musí být ze zařízení nejprve odebrána aplikace v uživatelském rozhraní OS, včetně uložených dat. Pokud byl již vytvořen adresář pro záznam videa v souborovém systém, je třeba jej vymazat.

Technologie Wi-Fi, 3G, Bluetooth i GPS jsou vypnuty.

#### 5.4.3.2 Spuštění aplikace a přihlášení

Po nahrání aplikace do zařízení je aplikace spuštěna. Na přihlašovací obrazovce je vyplněno reálné (na serveru existující) jméno a heslo, poté je kliknuto na přihlašovací tlačítko.

V dialogu vyžadujícím přístup k internetu je vybráno Wi-Fi nebo

3G. U obou voleb je následně ověřeno, zda došlo k aktivaci příslušné technologie v nastavení systému nebo systémové liště. Pokud ano, mělo by dojít po dalším odeslání formuláře k přihlášení.

Po přihlášení se v konzoli IDE kontroluje hlášení o stažení standardních PID kódů ze serveru. Zobrazená zpráva musí zobrazovat PID kódy v JSON formátu.

#### *5.4.3.3 Kontrola spojení OBD*

Po přihlášení do aplikace je vybráno v nastavení OBD zařízení (simulátor nebo OBD BT adaptér). Pokud zde není na výběr žádné zařízení, je třeba ověřit, zda je aktivován Bluetooth (pokud ne je to chyba, aplikace si BT aktivuje automaticky).

Po zvolení BT zařízení by si aplikace měla vyžádat restart. Po provedení restartu je buď vyžádáno nové provedení párování (to je v pořádku, nejedná se o chybu, neboť přerušení párování nemůže vyvolat aplikace) nebo rovnou dojde ke spojení s OBD.

Toto se projeví na hlavní obrazovce změnou jízdních údajů.

#### *5.4.3.4 Kontrola GPS*

Po přihlášení do aplikace je uživatel dotázán, zda chce zapnout GPS nebo si nepřeje službu používat. Vybereme „Nepoužívat GPS“, přejdeme do nastavení

aplikace do sekce GPS a ověříme, zda byla služba vypnuta. Pokud ano, ukončíme aplikaci.

Aplikaci znovu zapneme, tentokrát by se neměl dialog zobrazit. Poté GPS opět v nastavení aplikace aktivujeme. Aplikace vyžádá restart.

Po restartu klikneme na „Zapnu GPS“, pokud funguje přesměrování na nastavení systému do sekce GPS, aktivujeme GPS. Vrátíme se zpět do aplikace systémovým tlačítkem „zpět“ („<“).

Aplikace by měla spustit službu GPS. Zde je ale nutno mít na paměti, že během testu v reálném voze (tedy s reálným GPS) nedojde ke změně stavu služby na „OK“ ihned, neboť zaměření dostatečného počtu satelitů zabere nějaký čas. Stav služby lze mezi tím sledovat v konzoli IDE, pokud jsou zaznamenány alespoň nějaké satelity, lze při špatném počasí či slabém GPS signálu test ukončit (a považovat za úspěšný). V normálním případě by se měl změnit stav služby na „OK“ a v konzoli IDE by se měli zobrazit přesné souřadnice polohy.

### *5.4.4 Testovací jízdy*

Testovací jízdy byly prováděny mezi obcemi Pitkovice a Čestlice na východě Prahy. Byla vybrána referenční trasa s délkou 3 174 m, jež byla změřena na webu katastrálního úřadu [82].

#### 5.4.4.1 Kontrola jízdních údajů ve voze

Před provedením jízdy bylo obvykle již provedeno uživatelské testování, testovací jízdy byly tedy prováděny s verzí aplikace, kde již byla ověřena základní funkčnost. V aplikaci byl již také přihlášen uživatel k uživatelskému účtu.

Po nastartování vozu, byl ověřen stav komponent OBD a GPS. Připojení k internetu nebylo vyžadováno. Během stání byly zkontrolovány otáčky motoru oproti otáčkám na přístrojové desce vozidla. Nezobrazení nebo velké rozdíly mezi hodnotami byly považovány za chybu.

Při nezobrazení údaje bylo důležité vyzkoušet, zdali situaci nenapraví vysunutí a opětovné zasunutí OBD BT adaptéru do sběrnice, což bylo zaznamenáno jako problém u prototypu [3].

#### 5.4.4.2 Kontrola jízdních údajů po jízdě

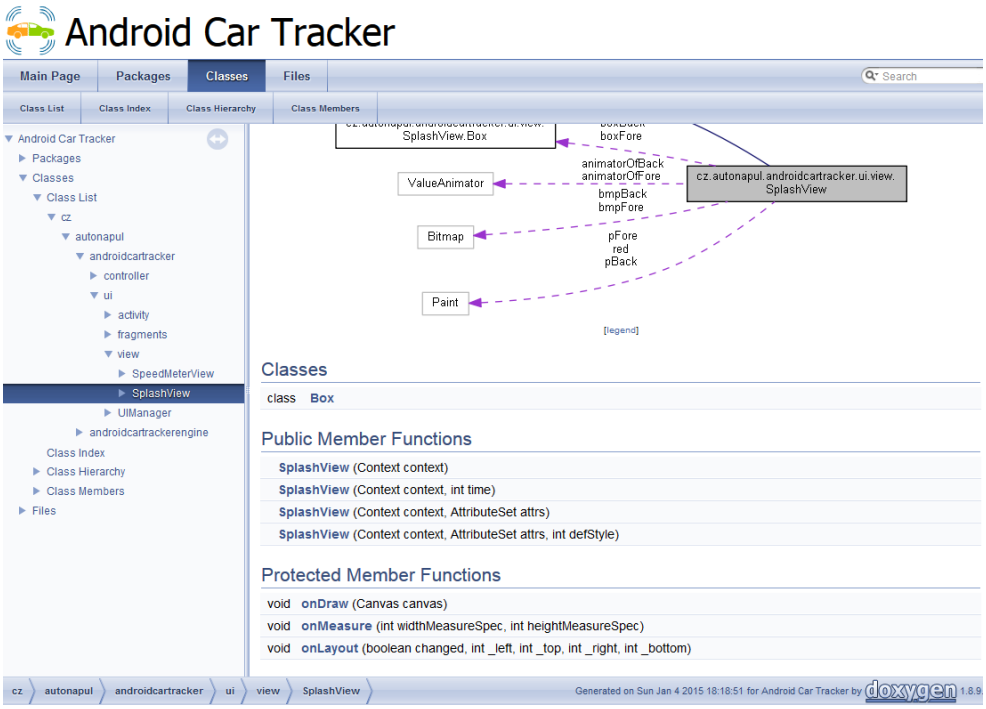
Vzhledem k tomu, že kontrolu údajů za jízdy nelze z hlediska bezpečnosti doporučit, byla pouze letmo srovnávána rychlost na tachometru s rychlostí získávanou z OBD a GPS. Velký rozdíl některé z hodnot vůči zbývajícím byl považován za chybu.

Po provedení jízdy byla provedena synchronizace (resp. odeslání) jízdních záznamů na server. Na serveru byla ověřena trasa jízdy na mapě a její shoda se skutečnou trasou. Ověřena byla korelace rychlostí z OBD a GPS a také celková trasa získaná integrací měřených hodnot.

## 5.5 Zdrojové kódy a dokumentace

Zdrojový kód aplikace (v jazyce JAVA [75]) je průběžně dokumentován a to jak na úrovni popisů tříd a metod, tak především uvnitř metod, což by mělo usnadnit orientaci programátorům, kteří budou projekt udržovat.

Programátorská dokumentace je také vygenerována jako statický web pomocí aplikace Doxygen [83]. Obsahuje rovněž diagramy tříd generované rozšířením Doxygen [84], pro lepší znázornění složité struktury aplikace.




*Obr. č. 31: Ukázka programové dokumentace*

Proměnné, metody a třídy jsou nazývány v angličtině, ve snaze co nejlépe popsat jejich funkci samotným názvem. Komentáře jsou potom psány česky s ohledem na komplexní problematiku, jež často vysvětlují.

# 6 Vývoj serverové aplikace

Vývoj serverové aplikace je čtvrtou a závěrečnou iterací iterativní přístupu k této práci. Řešena bude implementace a testování serverové aplikace, testování celého systému a implementace dodatečných vylepšení do obou aplikací.

## 6.1 Vývoj webu

Serverová aplikace byla vyvíjena nejprve lokálně, s použitím platformy XAMPP [10] verze 1.7.3., jež obsahuje HTTP [61] server Apache [13], interpret PHP [11] a databázi MySQL [59]. Později byla nasazena online na webhosting Savana.cz [12], kde jsou použity stejné technologie (kromě XAMPP).

Jako nástroj pro editaci zdrojových kódů nebylo použito IDE, pouze textový editor (Notepad++ [14]).

### *6.1.1 Abstrakce databáze*

Pro abstrakci databáze byl použit ORM framework RedBeanPHP [15] verze 3.5, jež je volně k dispozici pod BSD licencí [16]. Umožňuje vytváření objektů, jež odpovídají strukturou existujícím tabulkám databáze (viz kapitola 4.4.2).

Tento přístup je výhodný z hlediska jednoduchosti použití, neboť částečně (pro jednoduché dotazy nebo operace) odpadá potřeba práce s jazykem SQL [85].

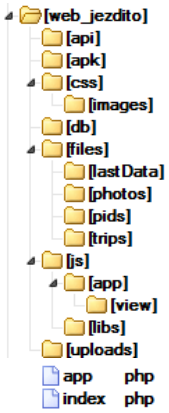
Přiklad použití abstrakce pro vytvoření záznamu uživatele v DB je uveden v následující ukázce (Obr. č. 32).:

```
$user = R::dispense('user');
$user->name = 'karel';
$user->password = md5('admin');
$id = R::store($user);
echo("User created with ID: ".$user->id);
```

*Obr. č. 32: Ukázka použití RedBeanPHP pro abstrakci DB*

### 6.1.2 Adresářová struktura

Vzhledem k faktu, že jízdní data mají nestrukturovanou podobu (nevíme přesně, jaká data dostaneme) budeme je ukládat ve formě souborů. Do databáze potom zaznamenáme pouze přítomnost souboru, časový záznam a jízdní vzdálenost.




*Obr. č. 33: Adresářová struktura serverové aplikace*

Data nahraná od uživatelů jsou ukládány do složky „/files“ jako soubory JSON, vyžádané fotografie z palubní jednotky potom jako obrázky JPG.

Pro visuální styly a JavaScript jsou vyhrazeny adresáře „/js“ a „/css“, jež jsou společné pro veřejnou i neveřejnou (pro přihlášené uživatele) část aplikace.

V adresáři „/api“ se nachází skripty PHP, jež slouží jako API, rovněž společné pro obě části serverové aplikace. V adresáři „/apk“ pak jsou umístěny instalovatelné balíčky palubní aplikace s příponou „.APK“.

### 6.1.3 API

Aplikační rozhraní serverové aplikace je použito nejen ke komunikaci s palubní jednotkou, ale také na uživatelské straně aplikace.

Liší se pouze způsob přihlášení. Zatímco palubní jednotka se přihlašuje identifikačním číslem („ID“) uživatele a před-generovaným klíčem, u uživatele přistupujícího do aplikace přes webový prohlížeč je vytvořeno tzv. „sezení“ (session [86]). V session jsou zaznamenány parametry přihlášeného uživatele, které slouží k identifikaci dalších požadavků stejného uživatele.

### 6.1.4 Použité JavaScript knihovny

V aplikaci je na mnoha místech použita knihovna jQuery [18] verze 1.11.1 a jQuery UI [19], oba volně dostupné pod MIT licencí [68].

Dále je na klientské straně použit framework BackboneJS [71] verze 1.1.2 společně s knihovnami:

- UnderscoreJS [72], verze 1.7.0
- RequireJS [73], verze 2.1.15
- JSON2 [74], verze 2014-02-04

Zmíněné knihovny jsou volně k dispozici pod MIT licencí [68], s výjimkou JSON2, jejíž zdrojový kód je sice otevřeně k dispozici, avšak licenční podmínky zde nejsou výslovně uvedeny.

### 6.1.5 Přihlašování a úvodní stránka

Mechanismus přihlašování a registrace je implementován za použití frameworku BackboneJS. Tento spolupracuje s PHP API „/api“, které provádí validaci zaslaných údajů a jejich ověření v databázi v případě, že se jeví v pořádku.

Přihlašování funguje stejným způsobem. Data se zasílají kódovaná v JSON notaci, a po úspěšném přihlášení je zahájena PHP session a uživatel je přesměrován na část webu dostupnou po přihlášení „app.php“.

Na úvodní stránce jsou dále zobrazeny základní informace k užívání systému a tlačítko ke stažení aplikace do zařízení uživatele.

Přední stránka je znázorněna na dalším obrázku (Obr. č. 34).

*Obr. č. 34: Úvodní webová stránka a registrace*

## 6.2 Zobrazení dat

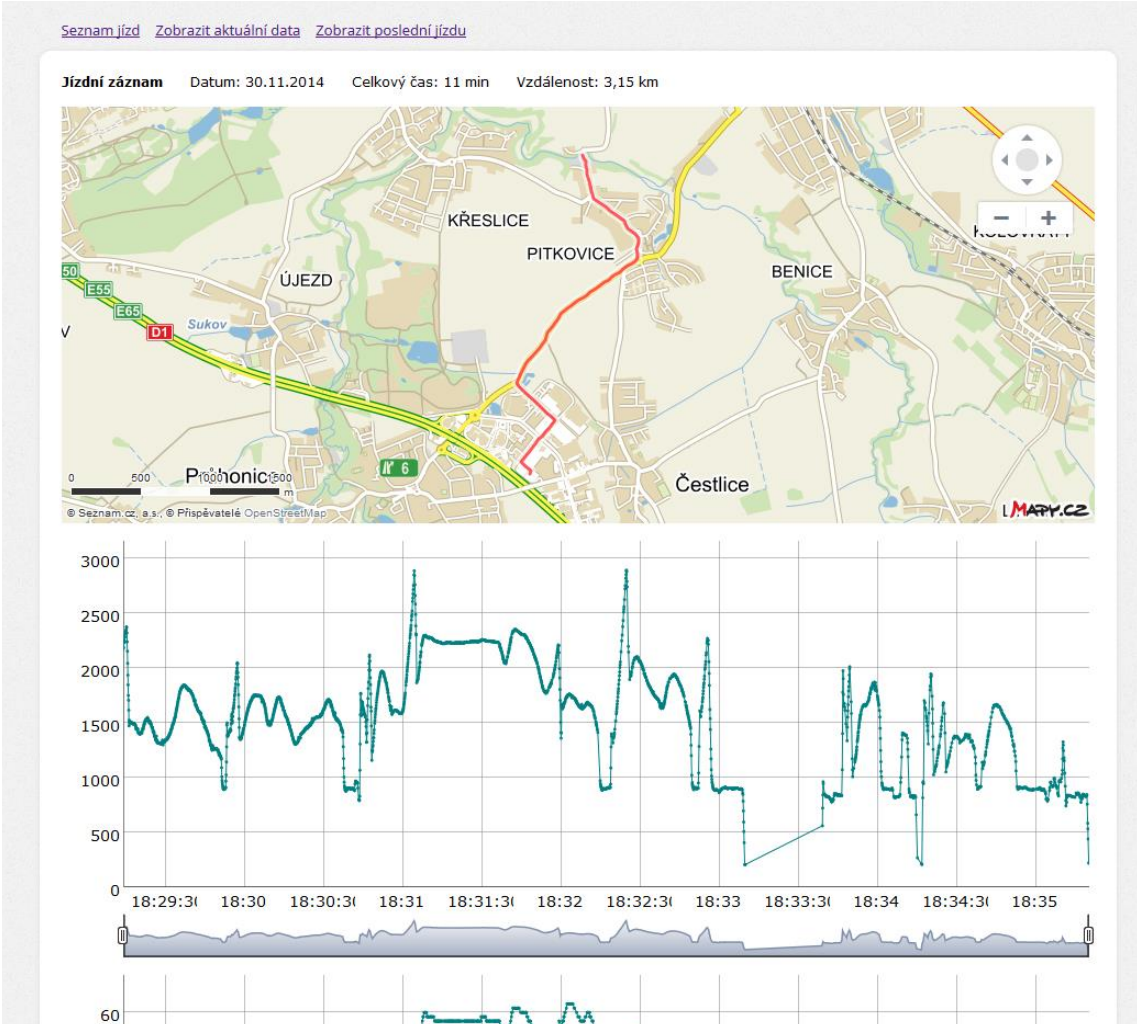
Pro zobrazení geografických dat bylo použito mapové API Mapy.cz [46], v aktuální verzi 4.8.

Pro zobrazení temporální dat (časových řad) ostatních zaznamenaných veličin je použito JS komponenty „dygraphs“ [17], jež je volně k dispozici pod MIT licencí [68].

Po přihlášení je uživateli nabídnuta stránka se seznamem jízd, jež náleží k jeho účtu dle databáze. Po výběru jízdy (nebo kliknutí na odkaz „Poslední jízda“) dojde k načtení příslušné jízdy prostřednictvím Require.JS. Po té co je načítání dokončeno stránka načte a vykoná soubor „tripToMap.js“, který provede vlastní zobrazení.

Zobrazení začíná základními údaji o jízdě a pokračuje mapou, kde je visuálně zaznamenána trasa jízdy (na Obr. č. 35 je zachycena referenční trasa). V mapě se není nutné na vyznačenou nikterak přibližovat nebo posouvat, pozice a přiblížení mapy je přepočteno ze souřadnic záznamů.

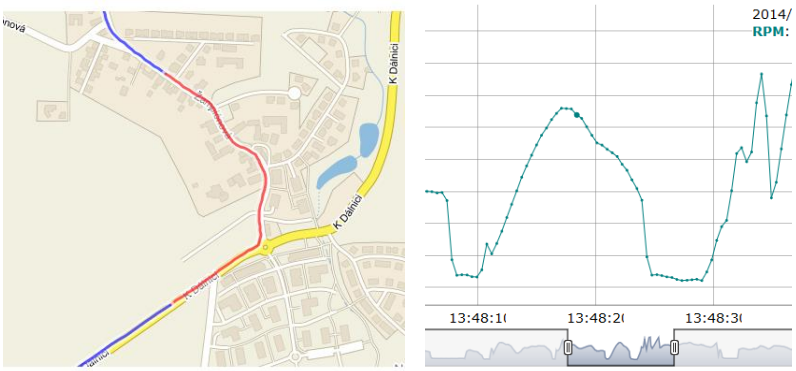
Následují grafy s jízdními údaji, dle definice příslušného JSON souboru.:




*Obr. č. 35: Zobrazení záznamu jízdy*

### 6.2.1 Vymezení úseku jízdy

Na grafech jízdních dat lze za pomocí posuvníků pod grafy vymezit jízdní úsek ke zobrazení (viz následující obrázek Obr. č. 36):




*Obr. č. 36: Vymezení úseku jízdy*

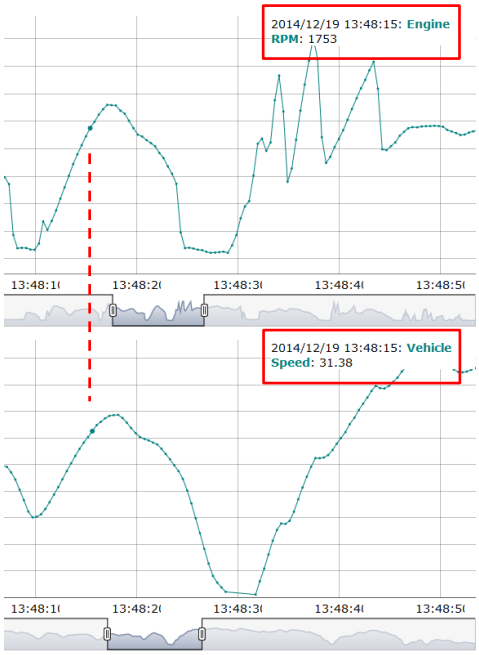
Vymezený úsek se okamžitě synchronizuje s mapou, kde se vybrané část jízdy zobrazí červenou barvou, zbytek trasy bude modrý. K synchronizaci dojde i mezi grafy jízdních veličin, k přiblížení na vybraný úsek dojde na všech zároveň.

### 6.2.2 Synchronizace datových bodů

K synchronizaci dochází také při najetí na graf, kdy se datový bod nejblíže ke kurzoru myši označí jako vybraný. Tento vybraný bod je pomocí své časové známky (timestamp) synchronizován s ostatními grafy, tak že i na ostatních grafech budou vybrány datové body se stejnou nebo ze všech nejpodobnější (nejbližší) časovou známkou.

Pro nalezení nejbližšího datového bodu v ostatních grafech je použit binární vyhledávací algoritmus [87]. Ten pracuje na principu půlení intervalů v poli seřazených hodnot, jakými jsou i naše záznamy jízdních dat.

Na následujícím obrázku jsou zvýrazněny datové body, synchronizované po najetí (Obr. č. 37).




*Obr. č. 37: Synchronizace datových bodů v grafech*

## 6.3 Další rozšíření

Během závěrečné iterace byly prozkoumány některé rozšiřující aspekty palubní jednotky i serverové aplikace, které by mohli v budoucnu přispět k větší použitelnosti celého systému.

### *6.3.1 GCM*

Palubní aplikace umožňuje online zasílání jízdních dat na server. Tato funkce ovšem vyžaduje určitou nezanedbatelnou režii, týkající se objemu přenesených dat. Vzhledem k faktu, že u některých telekomunikačních operátorů může být přenos dat zpoplatněn na základě přeneseného objemu dat, jistě se vyplatí hledat v této oblasti úspory.

Zde by mohla být nápomocna technologie GCM [88], jež byla implementována do palubní jednotky i serverové aplikace. Umožňuje vzdáleně kontaktovat spárovanou palubní jednotku a zaslat jí zprávu.

Není tedy již potřeba přístupu, kdy palubní aplikace musela zasílat data na server neustále. Pomocí GCM si můžeme data od palubní jednotky (či jednotek) přímo vyžádat.

V palubní jednotce je funkcionalita implementována ve třídě „GcmBroadcastReceiver“, na serveru potom PHP skripty „/api/gcmSend.php“ a „/api/gcmRegisterApp.php“.

Zařízení musí být pře použitím GCM spárována, což se automaticky provádí při prvním úspěšném přihlášení uživatele do palubní jednotky.

Technologie byla otestována a je plně funkční. Pokud má zařízení aktivováno 3G (a je v dosahu internetu), pak je zpráva vždy předána, a to dokonce i pokud je zařízení v režimu spánku, nebo aplikace není vůbec spuštěna (GCM zpráva vyvolá její spuštění). Pokud má zařízení pouze Wi-Fi připojení k internetu, pak je vyžadováno, aby pro příjem GCM zprávy zůstalo mimo režim spánku.

Tento postup byl také vyzkoušen pro vyvolání přenosu fotografií (zachycených z videokamery služby „VideoService“) z palubního zařízení na server, kde byly uloženy.

Zbývá již jen dodat, ke zprovoznění GCM (podrobněji zde [88]) je třeba registrovat příslušný vývojářský účet, povolit a nastavit zde využití technologie GCM.

### *6.3.2 NFC*

Vzhledem k původnímu určení aplikace pro car-sharing byla prozkoumána možnost využití technologie NFC pro identifikaci uživatele.

Byla zakoupena NFC karta typu Mifare 1K [89](Obr. č. 38) a byly na ni zaznamenány správné „ID“ a „upload_key“ existujícího uživatelského účtu, registrovaného na serveru.

Ve třídě „LoginActivity“ byla rozšířena implementace metody „onResume“ o kód nezbytný k dekódování obsahu karty. Přečtené údaje byly potom použity k úspěšnému přihlášení do aplikace (testováno na zařízeních Z3, Z4 a Z5, disponujících NFC).

Tato funkcionalita je kromě car-sharingové varianty (kde je nezbytná, neboť uživatel nemá klíče od vozu – používá jen kartu) velmi uživatelsky pohodlná i v osobní variantě. Odpadá ruční zadávání textových údajů do formuláře, jež se může omrzet hlavně v případě, že uživatel má více uživatelských účtů (pro více vozů).

*Obr. č. 38: NFC karta Mifare 1K*

### 6.3.3 PHP ZIP

Jízdní data jsou ukládána na serveru v podobě souborů. Znamená to tedy, že se je můžeme pokusit zkomprimovat.

Byla použita PHP třída „ZipArchive“ [90], pro kompresi pětice jízdních záznamů o velikosti 3340kB. Po kompresi na jednotlivé „.ZIP“ soubory zabírali společně pouze 843kB. To představuje úsporu téměř 75% prostoru, což je velmi příjemné s ohledem na fakt, že doba komprese nebyla nijak měřitelná (visuálně proběhlo zobrazení jízdních záznamů stejně rychle jako bez komprese).

## 6.4 Grafika a uživatelský dojem

Vzhledem k faktu, že projekt je ve zvolené variantě (viz kapitola 4.1.2) určený pro využití širokou veřejností, je vhodné vytvořit k projektu přívětivou a zapamatovatelnou visuální podobu.

### *6.4.1 Zdroje obrázků*

V tomto dokumentu byly využity pouze ikony ze stránky „iconarchive.com“ [94], jež jsou k využití zdarma pro nekomerční použití.

Při návrhu visuálních materiálů k projektu tyto ikony využity nebyly, byly použity pouze materiály nakreslené, vyfocené či vytvořené autorem práce s výjimkou siluet osob stažené z [95], které jsou k dispozici zdarma pro všechny druhy užití.

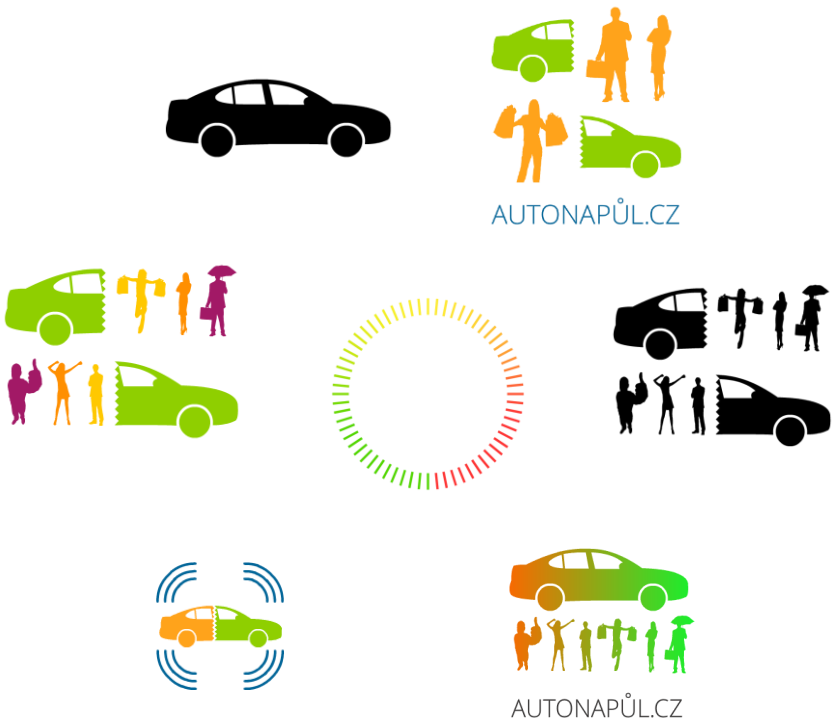
Grafické materiály byly zpracovány ve zkušebních verzích programů Adobe Photoshop CC [92] a Adobe Flash [93].

Diagramy v tomto dokumentu byly pro úplnost vytvořeny v software Enterprise Architect [96] a „draw.io“ [97], přičemž byly použity základní předdefinované tvary a objekty těchto aplikací.

### 6.4.2 Grafický návrh ikon a pozadí

Kromě webových stránek a visuální podoby aplikace, které již byly uvedeny v textu, bylo vytvořeno také několik dalších grafických materiálů, z nichž byla vybrána výsledná podoba pro použití v aplikaci.

Část z nich je pro odlehčení uvedena na následujícím obrázku:




*Obr. č. 39: Ukázka grafického návrhu projektu*

# 7 Výsledky a diskuse

## 7.1 Vyvinutý software

Podařilo se realizovat poměrně zajímavý systém Android aplikace a PHP serveru. Projekt může sloužit veřejnosti nejen ve své stávající podobě, ale může být modifikován a nabídnut také v jiných variantách (viz kapitola 4.1.2).

Použitými technologiemi a řešeními, ale i dokumentací a uvedenými ukázkami se práce snaží dláždit cestu pro další pokračování projektu Metrocar (viz kapitola 1.2) a podobných iniciativ.

## 7.2 Kritika

Ačkoli by projekt mohl být nasazen velmi brzy do provozu a jeho primární funkce by byly splněny, vzhledem k soukromému charakteru služby by bylo nejprve vhodné odstranit některé části, jež by mohli potenciálně narušit soukromí uživatelů (např. velmi rozsáhlé požadavky na práva aplikace v systému nebo možnost vzdáleného probuzení zařízení a pořízení fotografií).

Velmi zajímavým aspektem projektu, na který však již autorovy práce bohužel nezbyl čas, by byla podrobnější empirická analýza dat získaných z jízd s reálnými vozy, např. přesnost GPS, srovnání rychlostí z GPS a OBD, srovnání rozdílu mezi skutečnými a integrovanými jízdními vzdálenostmi apod.

## 7.3 Budoucí vývoj

### 7.3.1 Správa tankování

Pro osobní variantu osobnímu použití systému by bylo do budoucna jistě zajímavé uvažovat o správě tankování, jež by sloužila řidičům k evidenci nákladů za pohonné hmoty. Sebrané informace by byli jistě vhodným doplňkem stávající evidence jízd a projekt by tak získal z pohledu uživatele i ekonomický rozměr.

### 7.3.2 Odemykání vozu - lock

Vyřešení této problematiky je velmi vhodné pro další pokrok projektu ve variantě car-sharing (viz 4.1.2). Pro ni je nezbytně nutné, aby palubní jednotka dokázala odemknout či obecně fyzicky zpřístupnit vozidlo. Není totiž možné, aby měli všichni uživatelé fyzické klíče od všech vozů

Tato problematika byla diskutována již při vývoji prototypu [3]. V této práci byl učiněn další krok v podobě funkční a využitelné NFC autorizace uživatele. Otázka zpřístupnění vozů však přetrvává i nadále a lze ji doporučit jako vhodný námět na projekt s hardwarovým zaměřením (viz [3]).

### 7.3.3 USB NFC čtečka

Během testování NFC pro praktické využití v car-sharing variantě bylo zjištěno, že NFC čtečky zabudované v Android aplikacích mají typický dosah 55 mm (5.5cm). Tento se neukázal být ovlivněn materiálem skla automobilu. Avšak

díky typickému sklonu předního skla vozů je velmi složité nalézt vhodné místo pro uchycení zařízení takové, aby bylo možné účinně přiložit NFC kartu a načíst z ní informace.

Pro další vývoj v této oblasti lze tedy doporučit pouze použití externí USB NFC čtečky, přičemž se bude nutné při implementaci zabývat také nastolenou otázkou současného napájení zařízení (více v kapitole 4.3.1) jež byla v práci také řešena.

### *7.3.4 Více vozů pod jedním uživatelským účtem*

Pro majitele více vozů může být zajímavá možnost sdružit pod jedním uživatelským účtem více svých vozů. Podobná situace také může nastat pokud se o jedno vozidlo dělí více uživatelů, v takovém případě by byly jízdy vidět pouze pod uživatelským účtem uživatele, jež provedl jízdu a nikoli pod účtem majitele vozidla.

Řešením by mohlo být logické oddělení uživatelů a vozidel a vytvoření pomocného takového datového modelu na serveru i palubní jednotce, jež by toto nové členění umožňovalo.

# 8 Závěr

Tato práce se během čtyř iterací postupně zabývala analýzou projektu Metrocar se zaměřením na oblast sběru jízdních dat, návrhem na vylepšení serverové i palubní aplikace, a nakonec i jejich implementací a praktickým testováním.

Na základě zkušeností a poznatků předchozích řešitelů byly vytvořeny zcela nové aplikace pro sběr jízdních dat.

Palubní aplikace byla otestována na několika hostitelských zařízeních s různými verzemi OS a byla ozkoušena při použití se čtyřmi typy osobních vozů. Serverová aplikace byla nasazena, zpřístupněna online a její funkčnost byla také řádně otestována.

Celý systém byl ve všech testovaných konfiguracích funkční a je připraven ke zpřístupnění veřejnosti v režimu beta testu.

Během práce byla s úspěchem překonána řada technických problémů a výzev. I přesto, že se zaměření v průběhu práce mírně změnilo, podařilo se celkově naplnit a v některých ohledech i překonat stanovené cíle práce.

# Seznam obrázků

Obr. č. 1: Vývoj počtu uživatelů car-sharingových služeb ve světě, převzato z [1] 2

Obr. č. 2: Původní GUI pro zobrazení jízdních záznamů ..... 16

Obr. č. 3: Odpověď serveru na dotaz pro zobrazení jízdních dat ..... 17

Obr. č. 4: Jízdní obrazovka prototypu Android aplikace, převzato z [3] ..... 20

Obr. č. 5: Ukázka záznamu chyby z chybového logu serveru ..... 23

Obr. č. 6: Mapka variant systému dle využití ..... 27

Obr. č. 7: Use case diagram, případy použití systému ..... 28

Obr. č. 8: Graf zastoupení verzí OS Android na trhu, zdroj [44] ..... 33

Obr. č. 9: Graf rozšířenost serverových programovacích jazyků, zdroj [45] ..... 36

Obr. č. 10: Schéma systému pro sběr jízdních dat ..... 38

Obr. č. 11: Schéma zapojení palubní jednotky ..... 41

Obr. č. 12: Diagram systému služeb ..... 43

Obr. č. 13: Návrh přístrojové desky ..... 44

Obr. č. 14: Návrh přístrojové desky se zobrazeným menu ..... 45

Obr. č. 15: Návrh editačního dialogu PID ..... 46

Obr. č. 16: Návrh serverové aplikace – úvodní stránka ..... 48

Obr. č. 17: Návrh serverové aplikace – prohlížení jízd ..... 49

Obr. č. 18: Schéma databáze serverové aplikace ..... 50

Obr. č. 19: Definice vstupního bodu aplikace ..... 54

Obr. č. 20: Úvodní obrazovka palubní aplikace ..... 55

Obr. č. 21: Vysouvací menu šablony Navigation Drawer, zdroj [55] ..... 56

Obr. č. 22: Uživatelské dialogy ..... 57

Obr. č. 23: Obrazovky nastavení ..... 58

Obr. č. 24: Struktura projektu v IDE ..... 59

Obr. č. 25: Schéma systému služeb palubní aplikace. ..... 60

Obr. č. 26: Použití sběrnice událostí ..... 62

Obr. č. 27: Diagram služby a interpretu OBD zpráv ..... 65

Obr. č. 28: Diagram interpretace PID kódu ..... 66

Obr. č. 29: Adresářová struktura serverové aplikace ..... 80

# Seznam literatury

[1] **SHAHEEN, Susan A.; COHEN, Adam P.** Worldwide carsharing growth: An international comparison. *Institute of Transportation Studies*, 2008.

[2] **MARTIN, Elliot; SHAHEEN, Susan A.; LIDICKER, Jeffrey.** Impact of carsharing on household vehicle holdings. *Transportation Research Record: Journal of the Transportation Research Board*, 2010, 2143.1: 150-158.

[3] **KUBŮ, Roman.** Zpracování dat o provozu automobilů pro projekt Metrocar. 2013.

[4] **HANOUSEK, Jiří.** Lokalizace mobilních zařízení a její použití ve webových aplikacích. 2012.

[5] **WAGNER, Jan.** Příprava systému autonapul. cz na reálné nasazení. 2012.

[6] **JEČMÍNEK, Jakub.** Vývoj účetní komponenty do systému pro podporu sdílení aut. 2013.

[7] **POKORNÝ, Petr.** Vývoj geografického modulu pro systém carsharingové společnosti. 2013.

[8] **RIBEIRO, Vítor.** Mining Geographic Data for Fuel Consumption Estimation. 2012.

[9] **TICHY, Walter F.** A catalogue of general-purpose software design patterns. In: Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings. IEEE, 1997. p. 330-339.

[10] **XAMPP Apache + MySQL + PHP + Perl.** *XAMPP is the most popular PHP development environment.* [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *https://www.apachefriends.org/index.html*

[11] **PHP.** Hypertext Preprocessor. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://www.php.net/*

[12] **Webhosting Savana.cz.** Dobrý hosting pro vaše domény. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://www.savana.cz/*

[13] **The Apache.** HTTP Server Project. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://httpd.apache.org/*

[14] **Notepad++.** A free source code editor. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://www.notepad-plus-plus.org/*

[15] **RedBeanPHP 4.** The Power ORM. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://redbeanphp.com/*

[16] **BSD licence.** Wikipedia. [online]. 3.9.2014 [cit. 2014-12-14]. Dostupné z: *http://cs.wikipedia.org/wiki/BSD_licence*

[17] **Dygraphs.** Fast, flexible open source JavaScript charting library. [online]. 2014 [cit. 2014-12-14]. Dostupné z: *http://dygraphs.com/*

[18] **jQuery.** Write less, do more. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://www.jquery.com/*

[19] **jQuery UI.** jQuery user interface. [online]. © 2014 [cit. 2014-12-14]. Dostupné z: *http://jqueryui.com/*

[20] **Autonapůl.** *První český carsharing.* [online]. © 2014 [cit. 2014-10-18]. Dostupné z: *http://www.autonapul.org/*

[21] **Android.** *Android.* [online]. © 2014 [cit. 2014-8-20]. Dostupné z: *http://www.android.com/*

[22] **On-board diagnostics.** *Wikipedia.* [online]. 4.10.2014 [cit. 2014-10-17]. Dostupné z: *http://en.wikipedia.org/wiki/On-board_diagnostics*

[23] **Metrocar.** *Metrocar - Carsharing company.* [online]. © 2010 [cit. 2014-4-18]. Dostupné z: *http://autonapul.cz/*

[24] **Metrocar Jan Wagner.** *Assembla.* [online]. 7.10.2012 [cit. 2014-4-27]. Dostupné z: *https://www.assembla.com/spaces/wagnejan_metrocar/wiki*

[25] **Roští.cz.** *Smart hosting.* [online]. © 2014 [cit. 2014-4-22]. Dostupné z: *https://rosti.cz/*

[26] **PostgreSQL.** *PostgreSQL, the world's most advanced open source database.* [online]. © 2014 [cit. 2014-10-8]. Dostupné z: *http://www.postgresql.org/*

[27] **OBD-II PIDs.** *Wikipedia.* [online]. 10.4.2014 [cit. 2014-4-14]. Dostupné z: *http://en.wikipedia.org/wiki/OBD-II_PIDs*

[28] **Auto mini diagnostika ELM327 V1.5 OBD2 bluetooth.** *JShop.* [online]. © 2014 [cit. 2014-5-10]. Dostupné z: *http://www.j-shop.cz/cs/auto-moto/165-auto-mini-diagnostika-elm327-v15-obd2-bluetooth.html*

[29] **Bluetooth.** *Bluetooth Special Interest Group.* [online]. © 2014 [cit. 2014-5-10]. Dostupné z: *https://www.bluetooth.org/*

[30] **CAN bus.** *Wikipedia.* [online]. 5.4.2014 [cit. 2014-5-11]. Dostupné z: *http://en.wikipedia.org/wiki/CAN_bus*

[31] **Electronic control unit.** *Wikipedia.* [online]. 18.3.2014 [cit. 2014-5-11]. Dostupné z: *http://en.wikipedia.org/wiki/Electronic_control_unit*

[32] **OBD Interpreters – ELM327.** *ELM Electronics.* [online]. © 2014 [cit. 2014-5-19]. Dostupné z: *www.elmelectronics.com/obdic.html*

[33] **The Equipment and Tools Institute.** *ELM Electronics.* [online]. 2014 [cit. 2014-6-6]. Dostupné z: *http://www.etools.org/FullMembershipFAQ*

[34] **Python.** *Welcome to Python.org.* [online]. 2014 [cit. 2014-6-11]. Dostupné z: *https://www.python.org/*

[35] **Django.** The web framework for perfectionists with deadlines. [online]. © 2014 [cit. 2014-6-11]. Dostupné z: *https://www.djangoproject.com/*

[36] **OpenStreetMap.** OpenStreetMap. [online]. © 2014 [cit. 2014-6-22]. Dostupné z: *http://www.openstreetmap.org/about*

[37] **Android Developer Tools.** Adroid Developers. [online]. 2014 [cit. 2014-7-1]. Dostupné z: *http://developer.android.com/tools/help/adt.html*

[38] **Netbeans.** NetBeans IDE, The Smarter and Faster Way to Code. [online]. © 2013 [cit. 2014-7-14]. Dostupné z: *https://netbeans.org/*

[39] **Android 4.4 Compatibility Definition.** Android Compatibility Program. [online]. 27.11.2013 [cit. 2014-9-14]. Dostupné z: *https://static.googleusercontent.com/media/source.android.com/cs//compatibility/4.4/android-4.4-cdd.pdf*

[40] **Proximity sensor.** Wikipedia. [online]. 14.9.2014 [cit. 2014-9-14]. Dostupné z: *http://en.wikipedia.org/wiki/Proximity_sensor*

[41] **Wi-Fi Alliance.** Wi-Fi®: THE mobile technology. [online]. 14.9.2014 [cit. 2014-9-14]. Dostupné z: *http://www.wi-fi.org/*

[42] **3G.** Wikipedia. [online]. 31.8.2014 [cit. 2014-9-20]. Dostupné z: *http://en.wikipedia.org/wiki/3G*

[43] **What is API Level?.** Android Developers. [online]. 2014 [cit. 2014-9-20]. Dostupné z: *http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels*

[44] **Platform Versions.** Android Developers. [online]. 1.12.2014 [cit. 2014-12-6]. Dostupné z: *https://developer.android.com/about/dashboards/index.html#Platform*

[45] **Usage of server-side programming languages for websites.** Web Technology Surveys. [online]. 23.7.2014 [cit. 2014-9-20]. Dostupné z: *http://w3techs.com/technologies/overview/programming_language/all*

[46] **PHP vs. ASP.NET: Costs, Scalability and Performance.** Udemy blog. [online]. 27.11.2013 [cit. 2014-9-28]. Dostupné z: *https://www.udemy.com/blog/php-vs-asp-net/*

[47] **Mapy API.** Mapy.cz. [online]. © 2014 [cit. 2014-9-28]. Dostupné z: *http://api.mapy.cz/*

[48] **Google Maps API.** Google Developers. [online]. © 2014 [cit. 2014-9-28]. Dostupné z: *https://developers.google.com/maps/*

[49] **Google Maps/Google Earth APIs Terms of Service.** Google Developers. [online]. © 2014 [cit. 2014-9-28]. Dostupné z: *https://developers.google.com/maps/terms*

[50] **Communicating with the UI Thread.** Google Developers. [online]. 2014 [cit. 2014-10-4]. Dostupné z: *https://developer.android.com/training/multiple-threads/communicate-ui.html*

[51] **AsyncTask.** Random An Android Blog. [online]. 17.4.2014 [cit. 2014-10-4]. Dostupné z: *http://simonvt.net/2014/04/17/asynctask-is-bad-and-you-should-feel-bad/*

[52] **AsyncTask.** Android Developers. [online]. 12.8.2014 [cit. 2014-10-4]. Dostupné z: *http://developer.android.com/reference/android/os/AsyncTask.html*

[53] **Near field communication.** Android Developers. [online]. 9.10.2014 [cit. 2014-10-12]. Dostupné z: *http://developer.android.com/reference/android/os/AsyncTask.html*

[54] **Activity.** Android Developers. [online]. 12.8.2014 [cit. 2014-10-12]. Dostupné z: *http://developer.android.com/reference/android/app/Activity.html*

[55] **Navigation Drawer.** Android Developers. [online]. 2014 [cit. 2014-10-12]. Dostupné z: *https://developer.android.com/design/patterns/navigation-drawer.html*

[56] **Android APIs Reference, Package Index.** Android Developers. [online]. 20.12.2014 [cit. 2014-12-21]. Dostupné z: *http://developer.android.com/reference/packages.html*

[57] **Menus.** Android Developers. [online]. 2014 [cit. 2014-10-12]. Dostupné z: *http://developer.android.com/guide/topics/ui/menus.html*

[58] **Android Studio, The Official Android IDE.** Android Developers. [online]. 2014 [cit. 2014-12-13]. Dostupné z: *http://developer.android.com/sdk/index.html*

[59] **MySQL.** The world's most popular open source database. [online]. © 2014 [cit. 2014-10-18]. Dostupné z: *http://www.mysql.com/*

[60] **JSON.** Introducing JSON. [online]. 2014 [cit. 2014-10-18]. Dostupné z: *http://www.json.org/*

[61] **Hypertext Transfer Protocol.** Wikipedia. [online]. 16.10.2014 [cit. 2014-10-25]. Dostupné z: *http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol*

[62] **Layouts.** Android Developers. [online]. 2014 [cit. 2014-11-2]. Dostupné z: *http://developer.android.com/guide/topics/ui/declaring-layout.html*

[63] **App Manifest.** Android Developers. [online]. 2014 [cit. 2014-11-2]. Dostupné z: *http://developer.android.com/guide/topics/manifest/manifest-intro.html*

[64] **Intents and Intent Filters.** Android Developers. [online]. 2014 [cit. 2014-11-2]. Dostupné z: *http://developer.android.com/guide/components/intents-filters.html*

[65] **Gradle.** Build Automation Evolved. [online]. © 2014 [cit. 2014-11-8]. Dostupné z: *https://www.gradle.org/*

[66] **Singleton pattern.** Wikipedia. [online]. 5.11.2014 [cit. 2014-11-9]. Dostupné z: *http://en.wikipedia.org/wiki/Singleton_pattern*

[67] **MBassador.** GitHub. [online]. © 2014 [cit. 2014-11-9]. Dostupné z: *https://github.com/bennidi/mbassador*

[68] **The MIT License (MIT).** Open Source Initiative. [online]. 2014 [cit. 2014-11-9]. Dostupné z: *http://opensource.org/licenses/MIT*

[69] **Annotations Basics.** ORACLE Java Documentation. [online]. © 2014 [cit. 2014-11-9]. Dostupné z: *http://docs.oracle.com/javase/tutorial/java/annotations/basics.html*

[70] **Bluetooth.** Android Developers. [online]. 2014 [cit. 2014-11-15]. Dostupné z: *http://developer.android.com/guide/topics/connectivity/bluetooth.html*

[71] **BackboneJS.** Gives structure to web application. [online]. 2014 [cit. 2014-12-20]. Dostupné z: *http://backbonejs.org/*

[72] **UnderscoreJS.** [online]. 2014 [cit. 2014-12-20]. Dostupné z: *http://underscorejs.org/*

[73] **RequireJS.** A javascript module loader. [online]. © 2014 [cit. 2014-12-20]. Dostupné z: *http://requirejs.org/*

[74] **JSON-js, JSON in JavaScript.** GitHub. [online]. 18.11.2010 [cit. 2014-12-20]. Dostupné z: *https://github.com/douglascrockford/JSON-js*

[75] **Java SE at a Glance.** Oracle. [online]. 2014 [cit. 2014-12-20]. Dostupné z: *http://www.oracle.com/technetwork/java/javase/overview/index.html*

[76] **Java SE at a Glance.** Oracle. [online]. 2014 [cit. 2014-12-20]. Dostupné z: *http://www.oracle.com/technetwork/java/javase/overview/index.html*

[77] **Class Thread.** Java™ Platform SE 7. [online]. © 2014 [cit. 2014-12-20]. Dostupné z: *http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html*

[78] **API Specification.** Java™ Platform SE 7. [online]. © 2014 [cit. 2014-12-20]. Dostupné z: *http://docs.oracle.com/javase/7/docs/api/overview-summary.html*

[79] **Saving Key-Value Sets.** Android Developers. [online]. 2014 [cit. 2014-12-20]. Dostupné z: *http://developer.android.com/training/basics/data-storage/shared-preferences.html*

[80] **Model–view–controller.** Wikipedia. [online]. 9.12.2014 [cit. 2014-12-20]. Dostupné z: *http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller*

[81] **OBDSim.** OBD GPS Logger. [online]. 2014 [cit. 2014-12-20]. Dostupné z: *http://icculus.org/obdgpslogger/obdsim.html*

[82] **Nahlížení do katastru nemovitostí.** Státní správa zeměměřictví a katastru. [online]. © 2014 [cit. 2014-12-20]. Dostupné z: *http://nahlizenidokn.cuzk.cz/*

[83] **Doxygen.** Generate documentation from source code. [online]. 21.8.2014 [cit. 2014-12-20]. Dostupné z: *http://www.stack.nl/~dimitri/doxygen/*

[84] **Graphviz.** Graph Visualization Software. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *http://www.graphviz.org/*

[85] **SQL Tutorial.** W3Schools. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *http://www.w3schools.com/sql/*

[86] **Session Handling.** PHP. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *http://php.net/manual/en/book.session.php*

[87] **Binární vyhledávání.** Algoritmy.net. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *http://www.algoritmy.net/article/21/Binarni-vyhledavani*

[88] **Google Cloud Messaging for Android.** Android Developers. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *https://developer.android.com/google/gcm/index.html*

[89] **Plastová NFC karta Mifare 1K.** NFCmix. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://www.nfcmix.com/cs/nfc-karty-vizitky/93-karta-plastova-mifare-1k-cr80.html*

[90] **ZIP.** PHP. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://php.net/manual/en/book.zip.php*

[91] **ZIP.** PHP. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://php.net/manual/en/book.zip.php*

[92] **Adobe Photoshop family.** Adobe. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://www.adobe.com/en/products/photoshopfamily.html*

[93] **Flash Professional CC.** Adobe. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://www.adobe.com/en/products/flash.html*

[94] **Icon Archive.** Search through more than 558000 free icons. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://www.iconarchive.com/*

[95] **GFX9.COM.** Set of people silhouettes vector. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *http://gfx9.com/set-of-people-silhouettes-vector/151416*

[96] **UML Modeling and Lifecycle Tool Suite.** Sparx Systems. [online]. © 2014 [cit. 2014-12-27]. Dostupné z: *http://www.sparxsystems.com.au/*

[97] **Draw.io.** Free online diagram software. [online]. 2014 [cit. 2014-12-27]. Dostupné z: *http://www.sparxsystems.com.au/*

# Seznam použitých zkratek

**3G** Third generation of mobile telecommunications technology, telekomunikační technologie umožňující přesnost hlasu a dat

**ADT** Android Development Tools, IDE pro OS Android

**API** Application Programming Interface

**ASP** Active Server Pages

**BT** Bluetooth

**CAN** Controller Area Network, sběrnice vnitřní komunikační sítě

**DB** DataBase (databáze)

**ECU** Electronic Control Unit (elektronická řídící jednotka, nebo jen „řídící jednotka“)

**FTP** File Transfer Protocol

**GCM** Google Cloud Messaging

**GPS** Global Position System (globální polohový systém), systém pro určování polohy

**GUI** Graphical User Interface (grafické uživatelské rozhraní)

**HTTP** HyperText Transfer Protocol, internetový protokol

**IDE** Integrated Development Environment (integrované vývojové prostředí)

**JPEG** Joint Photographic Experts Group, formát obrázků

**JS** JavaScript

**JSON** JavaScript Object Notation

**MVC** Model–View–Controller

**NFC** Near Field Communication

**OBD** On-Board Diagnostic (palubní diagnostika)

**ORM** Object-Relational Mapping (objektově relační zobrazení)

**OS** Operating System (operační systém)

**OSM** Open Street Map

**PHP** Hypertext Preprocessor

**PID** Parameter ID (identifikace parametru), kód sloužící k identifikaci OBD dotazu

**RFID** Radio Frequency Identification (identifikace na radiové frekvenci)

**SQL** Structured Query Language

**UI** User Interface (uživatelské rozhraní)

**URL** Uniform Resource Locator (jednotná adresa zdroje)

**USB** Universal Serial Bus (universální sériová sběrnice)

**Wi-Fi** a.k.a. **WLAN** Wireless Local Area Network

**XML** Extensible Markup Language

# Obsah přiloženého CD

/ (kořenový adresář)

+ jungmto1_dp.docx - bakalářská práce ve formátu DOCX

+ jungmto1_dp.pdf - bakalářská práce ve formátu PDF

+ prirucka.pdf – uživatelská příručka Jezdíto.cz

/ android - adresář se zdroj. kódy pro platformu Android

/ AndroidCarTracker – aplikace palubní jednotky

/ AndroidCarTrackerEngine – knihovní modul systému služeb

/ php - adresář se zdroj. kódy serverové části pro platformu PHP

/ web_jezdito – webový frontend serveru
/ web_app – webový backend serveru

/ java - adresář se zdrojovými kódy propojovací knihovny

/ AjaxTester – utilita k simulaci http dotazu
/ FormulaParser – oddělená součást interpretu kódů PID