ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

Ústav dopravní telematiky K620

Návrh softwaru pro podporu realizace dopravních průzkumů

Software Design of Traffic Survey Realization Support

Diplomová práce

Studijní program: Technika a technologie v dopravě a spojích

Studijní obor: Inženýrská informatika v dopravě a spojích

Vedoucí práce: Ing. Martin Langr

Bc. Ivan Boyarkin

Praha 2015

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
d ě k a n
Konviktská 20, 110 00 Praha 1

K620..... Ústav dopravní telematiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Ivan Boyarkin

Kód studijního programu a studijní obor studenta:

N 3710 – ID – Inženýrská informatika v dopravě a spojích

Název tématu (česky): **Návrh softwaru pro podporu realizace dopravních průzkumů**

Název tématu (anglicky): Software Design of Traffic Survey Realization Support

Zásady pro vypracování

Při zpracování diplomové práce se řiďte osnovou uvedenou v následujících bodech:

- Analýza požadavků na software
- Analýza a volba vhodného nástroje pro tvorbu softwaru
- Návrh a vývoj softwaru
- Ověření funkčnosti realizovaného návrhu

Rozsah grafických prací: dle pokynů vedoucího DP

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury: Richardson, A. J., Ampt, E. S., Meyburg, A. H.; Survays methods for transport planning; Eucalyptus press, 1995

dokumentace vývojového prostředí zvoleného v analytické části práce

Vedoucí diplomové práce: **Ing. Martin Langr**

Datum zadání diplomové práce: **31. července 2014**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **31. května 2015**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

L. S.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní

doc. Ing. Pavel Hrubeš, Ph.D.
vedoucí
Ústavu dopravní telematiky

prof. Dr. Ing. Miroslav Svítek
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Ivan Boyarkin
jméno a podpis studenta

V Praze dne 31. července 2014

Poděkování

Na tomto místě bych rád poděkoval všem, kteří mi poskytli podklady pro vypracování této práce. Zvláště pak děkuji Ing. Martinu Langrovi za odborné vedení a konzultování diplomové práce a za rady, které mi poskytoval po celou dobu mého studia a dále bych ctěl poděkovat kolegům z projektu „Řízení a modelování silniční dopravy“ a všem, jejichž rady přispěly velkou měrou ke zpracování této diplomové práce.

V nepolední řadě je mou milou povinnosti poděkovat svým rodičům za morální a materiální podporu, které se mi dostávalo po celou dobu studia.

Práce vznikla za podpory grantu: SGS15/105/OHK2/1T/16.

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci na závěr studia na ČVUT v Praze Fakultě dopravní.

Nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).“

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

V Praze dne 31. května 2015 …….

Podpis

ČESKÉ VYSOKÉ ÚČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

NÁVRH SOFTWARU PRO PODPORU REALIZACE DOPRAVNÍCH PRŮZKUMŮ

diplomová práce
květen 2015
Bc. Ivan Boyarkin

ABSTRAKT

Předmětem diplomové práce „Návrh softwaru pro podporu dopravních průzkumu“ je zjištění způsobů získávání charakteristik dopravních proudu, analýza požadavků, které jsou kladeny na nástroje pro dopravní průzkumy a na základě této analýzy navrhnout a vyvinout aplikaci pro chytrá mobilní zařízení a zjistit, jestli je použitelná pro nahrazení papírových formulářů

Klíčová slova: dopravní průzkum, měření, návrh softwaru, vývoj softwaru, Android, mobilní operační systémy, chytré telefony.

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Transportation Sciences

SOFTWARE DESIGN OF TRAFFIC SURVEY REALIZATION SUPPORT

master‘s thesis
May 2015
Bc. Ivan Boyarkin

ABSTRACT

The subject of the master’s thesis „Software Design of Traffic Survey Realization Support” is finding ways to obtaining the traffic flow characteristics, analyzing requirements for a traffic survey tools and based on this analysis, design and develop application for smart mobile devices and find out if it is applicable to replace traffic survey sheets.

Keywords: traffic survey, measurement, software design, software development, Android, mobile operation systems, smartphone.

Obsah

Seznam použitých zkratek

APK	typ spustitelného souboru pro instalaci aplikace, které určeno pro OS Android
Bluetooth®	proprietární otevřený protokol pro bezdrátovou komunikaci
Cloudové služby	služby na internetu, které jsou poskytovány uživatelům odkudkoliv
CSV	Comma-Separated Values - hodnoty oddělené čárkami
DTD	Document Type Definition - definice typu dokumentu
Fishing	získávání citlivých údajů na internetů od obětí útoku
GNSS	Global Navigation Satellite Systém - globální družicový polohový systém
GNU	operační systém složený ze zcela svobodného softwaru
GNU GPL	GNU General Public License - všeobecná veřejná licence GNU
GPS	Global Positioning System – globální družicový polohový systém provozování Ministerstvem obrany Spojených států amerických
IDE	Integrated Development Environment – vývojové prostředí
JSON	JavaScript Object Notation - JavaScriptový objektový zápis
KML	Keyhole Markup Language – značkovací jazyk Keyhole
LOS	Level of service – úroveň služby
Malware	počítačový program pro vniknutí a poškození operačního systému
NAND	Not AND – negovaný logický součin
NFC	Near Field Communication - technologie radiové bezdrátové komunikace na krátkou vzdálenost
OBU	OnBoard Unit – palubní jednotka
OS	operační systém
POS terminál	Point-of-sale – platební terminál
RAID	Redundant Array of Inexpensive/Independent Disks - vícenásobné diskové pole laciných/nezávislých disků
SDK	Software Development Kit – soubor nástrojů pro vývoj software
Solid-state media nebo **SSD**	Typ datového media, který neobsahuje pohyblivé části
SQLite	relační databázový systém
TP	Technické podmínky
Unicode	tabulka znaků všech existujících abeced
USB	Universal Seriál Bus - univerzální sériová sběrnice
UTF-8	druh kódování ve standardu Unicode

W3C	World Wide Web Consortium – mezinárodní konsorcium vyvíjející webove standardy
WYSIWYG	způsob editace dokumentu, při kterém obsah dokumentu v procesu editace je zobrazen totožně s výslednou verzi.
XML	eXtensible Markup Language - rozšiřitelný značkovací jazyk
YAML	formát pro serializaci strukturovaných dat

1 Úvod

Dopravní průzkumy tvoří nedílnou část jakékoliv studie při návrhu dopravní infrastruktury. Nesmíme podceňovat to, jak jsou důležité pro organizaci dopravy a výstavbu nových komunikací. Je potřeba dbát nejenom na rozsah měření, ale i na kvalitu. Jelikož i já jsem se osobně setkal s problémem, ke kterému došlo při bodovém měření, způsobeným chybou určení přesného času. Může se zdát, že několik vteřin nic neovlivní, ale právě špatná synchronizace hodin na jednotlivých stanovištích vyvolala kolaps celého měření. Zabránit výskytu podobných situací mohou rychle rozvíjící se technologií spolu s cenovou dostupnosti osobní výpočetní techniky přenosné velikosti. Zamezit vlivu lidského faktoru zcela nedokážeme, ale můžeme jeho minimalizovat. Mimo jiné lze dosáhnout určitého zjednodušení, které by mohlo nejenom zmírnit zátěž na osoby vykonávající měření ale i snížit požadavky na lidské zdroje.

Téma diplomové práce jsem si zvolil v návaznosti na vlastní bakalářskou práci, která je věnovaná problematice použití dotykových technologií pro dopravní průzkumy. Její výsledky budou použity v následujících částech. Nejpodstatnější je, že z cenové dostupnost a převládání jenom jediné technologií na trhu ovlivnily celkové zaměření na segment chytrých mobilních zařízení.

Nesmím opomenout ani vlastní zkušeností s dopravními průzkumy a osobně můj názor na to, že při krátkodobých měření za pomocí lidské sily je obrovský prostor pro realizaci a implementaci různých nástrojů, které by přispěly k zjednodušení a zefektivnění činnosti. Zaznamenávat průjezdy vozidel, chodci nebo jakýkoliv jiný výskyt události na papír a pomocí tužky je klasický způsob, ale když žijeme v tak moderní době, je vhodné se pokusit navrhnout chytřejší nástroje pro dopravní průzkumy.

Z velkého množství způsobu získávání charakteristik dopravního proudu práce je zaměřená na manuální, neboli ruční měření. Zdůvodnění a seznámení s takovým výběrem je popsáno v další kapitole. Právě s těmito druhy průzkumů nejčastěji se setkávají studenti fakulty dopravní. A jsou nejnáročnější pro osoby, které měření vykonávají hlavně kvůli potřebě přepisu dat z papírových formulářů nebo z videa do elektronické podoby. Proto svoji prací bych viděl jako příspěvek ke zkvalitnění dopravních průzkumů, který budou moci využívat studenti naši fakulty.

Za očekávaný cíl jsem pokládal realizaci aplikaci pro chytrá přenosná zařízení, která by nahradila papírové záznamy, a byl by vynechán jeden nutný přepis dat do elektronické podoby.

2 Analýza požadavků na software

Před tím, než začnu navrhovat aplikaci, je nezbytné stanovit požadavky. Umožní to zabránit marnění času při vývoje funkci, které nebudou využity. Je zapotřebí přesně vědět, které nároky musí být splněny a co všechno software musí umět. Proto tato kapitola je věnovaná především dopravním průzkumům a způsobům získávání charakteristických hodnot dopravních proudů za účelem stanovení vlastností, kterých musí nabýt aplikace.

2.1 Dopravní průzkumy

V roce 2013 v Praze byly dokončeny výstavby více než 3000 bytových jednotek [1]. Staví se nová sídliště, lidí používají osobní vozidla, veřejnou dopravu. Pokud si uvědomíme počet osobních vozidel v jedné domácnosti a to všechno vyvolává novou poptávku a klade nové zvyšující se nároky na infrastrukturu. Celou situaci komplikuje stochastický původ dopravních proudů, především kvůli vlivu lidského faktoru. Vzniklé neurčitostí v přepravních procesech potřebujeme odstraňovat pomocí stálého pozorování a získávání charakteristik a vlastností dopravních proudu.

2.1.1 Dopravní proudy a jejích charakteristiky

Dopravní proud je obecně složen z různorodých vozidel pod řízením řidičů s odlišnými zkušenostmi, a která májí rozlišné dynamické vlastností, což určuje heterogennost dopravních proudů. V makroskopickém měřítku je možné pozorovat podobnost chování vůči proudění tekutin. Při dosazení maximální propustností určitého úseku, ten se začne postupně zaplňovat vozidly a některá z nich budou přetékat do vedlejších ulic.

V mikroskopickém měřítku chování jednotlivých vozidel ovlivněno nejenom dopravními předpisy. Velký vliv má dokonce i psychický stav řidiče a samozřejmě nesmíme zapomenout na technický stav vozidla nebo počasí. Ve výsledku předpovídaní pohybu a interakce dopravních prostředku není tak jednoduchou záležitosti, jak by se to mohlo zdát na první pohled.

2.1.1.1 Intenzita

Intenzita je důležitým parametrem dopravního proudu a obvykle definovaná jako počet vozidel projíždějících jedním bodem za určitý časový okamžik (běžně se používá jedna hodina). Je vyjádřena následujícím vztahem:

$$q = \frac{N}{T} \tag{1}$$

kde: q – intenzita $[voz/h]$

N – počet vozidel, která projela určitým profilem $[voz]$

T - délka trvání měření $[h]$

Volbu vhodného času pro měření intenzity dopravního proudu je zapotřebí pečlivě uvažovat. A to nejenom z hlediska trvání, ale i z hlediska hodiny ve dne, dne v týdnu, měsíce v roce atd. Jelikož tento parametr se liší jenom dle toho, jestli měření se provádí v pracovní den nebo o víkendu. Denní intenzity jsou ovlivněny polohou měřicího stanoviště: např. centrům města, průmyslová zóna a rekreační středisko budou mít jiný vývoj intenzit během dne.

Charakteristiky dopravních proudu na určitých úsecích jsou závislé taktéž na dni v týdnu. Především rozdíl je vidět mezi komunikacemi nacházejícími se v intravilánu a extravilánu. Městské komunikace jsou nejvíce zatížené během pracovních dnů a o víkendu, svátcích nebo prázdninách zaznamenávají pokles intenzit. Mimoměstské silnice v pracovní dny jsou vytížené méně.

Pro zjištění charakteru provozu na komunikaci II. a III. třídy se používá tzv. nedělní faktor. Pro jeho výpočet je zapotřebí data o intenzitě dopravy z pracovního dne a z neděle tj. je nutné provést dva průzkumy. S použitím nedělního faktoru lze rozdělit komunikace do třech skupin, které jsou uvedeny v tabulce Tabulka 1, a jeho hodnotu lze spočítat pomocí vztahu (2). [2]

$$f_{Ne} = \frac{I_{Ne,16-20}}{I_{BPD,13-17}}, \qquad (2)$$

kde: f_{Ne} nedělní faktor $[-]$

$I_{Ne,16-20}$ intenzita dopravy v běžnou neděli v době 16:00 – 20:00 $[voz/4h]$

$I_{BPD,13-17}$ intenzita dopravy v běžný pracovní den v době 13:00 – 17:00 $[voz/4h]$

Tabulka 1. Rozdělení komunikací do skupin podle charakteru provozu

Charakter provozu	Popis	f_{Ne} $[-]$
Hospodářský (H)	Komunikace je ve větší míře se používá pro pravidelné dojíždění do zaměstnání a škol v pracovní dny. Rekreační cesty jsou zastoupeny v menší míře.	$< 0{,}85$
Smíšený (S)	Komunikace se používá pro pravidelné cesty v pracovní dny tak i o víkendu. Provoz je rovnoměrný po celý týden.	$0{,}85 - 1{,}20$
Rekreační (R)	Na komunikaci převládají především cesty do rekreačních oblastí. Nárůst intenzit v páteční odpoledne směrem do rekreačních oblastí a v neděli směrem z rekreačních oblastí.	$> 1{,}20$

Použití nedělního faktoru není univerzální a je omezeno pouze komunikacemi v blízkosti hraničních přechodu, velkých zdrojů a cílů se specifickým chováním. V ostatních případech je nutné buď zvážit charakter provozu (tyká se účelových komunikací) nebo použit speciální přepočtové koeficienty (např. komunikace v blízkosti obchodních středisek).

Pohyb jednotlivých vozidel, ze kterých se sestávají dopravní proudy, v průběhu dne má různý charakter. Ráno lidí dojíždějí do zaměstnání, do škol atp. Zde se začíná ranní dopravní špička, která přetrvává několik hodin a dosahuje svého maxima kolem osmé hodiny ranní. Následně nastává mírný pokles, a který se kolem poledne změní na nárůst, kde postupně dostává do odpolední špičky.

Z výšeuvedených faktorů vyplývá, že pro stanovení přesné hodnoty intenzity dopravy je nutné provádět průzkumy nepřetržitě během určité sledované časové doby. Jelikož není to ve většině případů možné (je omezeno především lidskými zdroji), je zapotřebí hodnoty získané z kratší doby měření spočítat odhady intenzit, které bohužel jsou zatížené chybami a jejích přesnost je ovlivněná dobou průzkumu a charakterem provozu na komunikaci.

Doba průzkumu ovlivňuje výsledné hodnoty podle toho, kolik měření časově trvalo. Čím déle trval průzkum, tím přesnější hodnotu odhadu lze obdržet ve výsledku. Přesnost je závislá na podílu naměřené hodnoty intenzity dopravy za dané období z celkové intenzity dopravy.

Pokud charakter provozu na měřeném úseku komunikaci je více podoben charakteristickým průběhům intenzit na dané kategorii komunikaci - přesnost odhadu je vyšší.

Předpis ministerstva dopravy Technické podmínky č. 189 od 6. června 2012 stanoví vztah, podle kterého lze spočítat hodnotu odchylky:

$$\delta = 0{,}95 \cdot \left(\frac{I_m}{RPDI} \cdot 100\right)^{-0{,}60}, \qquad (3)$$

kde: δ odchylka odhadu ročního průměru denních intenzit [%]
I_m intenzita doprava v době průzkumu [voz/doba průzkumu]
$RPDI$ odhad ročního průměru denních intenzit dopravy

Dosáhnout nižších hodnot odchylek při vypočtu odhadu a tím i získat lepší výsledek lze pomocí několika způsobů [2], které se spočívají především ve zkvalitňování vzorku získaného měřením:

- opakování měření se změnou nejenom dne v týdnu, ale i časové doby, během které probíhá průzkum. Přičemž se musí volit přibližně podobné dny, tj. v případě pracovních

dnů se doporučuje volit další pracovní den, ale nikoliv den víkendový. Aplikace daného způsobu umožňuje získat během několikadenního měření přesnosti kolem 6%,

- volba vhodnější doby měření s největší intenzitou z hlediska podílu naměřené hodnoty na odhadu RPDI,
- pomocí dlouhodobého sledování zjistit průběh změn intenzit během týdnu a roku.

2.1.1.2 Hustota

Hustota je dalším důležitým parametrem charakteristiky dopravního proudu. Je definována jako počet vozidel na jednotku délky [voz/km] nebo jako počet vozidel na jednotku délky v jednom pruhu [voz/km/pruh]. Získávání hodnot hustoty přímo z dopravního proudu je obtížně realizovatelné pomoci běžně používaných přístupu a detektorů. Ale jelikož je důležitým parametrem pro zkoumání kongescí, obvykle se vypočítává z vedlejších veličin, jako např. převracená hodnota délkového rozestupu mezi čely vozidel:

$$D = \frac{1000}{d_d} \tag{4}$$

kde: D - hustota, $[voz/km]$

d_d - střední délkový rozestup, $[m]$

Jednou ze základních vlastností hustoty je její klesající závislost na rychlosti. Hustota klesá se zvětšující se rychlosti. Je to dáno dodržováním bezpečné vzdáleností mezi vozidly. Pokud budeme vycházet z doporučeného dvouvteřinového rozestupu na jednopruhové komunikace o délce jeden kilometr, a uvažovaní délky osobního vozidla v průměru $4{,}72\ m$ [3], získáme hodnoty obsazené v tabulce Tabulka 2

Tabulka 2. Ukázka klesající závislosti hustoty na rychlosti

Rychlost [km/h]	Bezpečná vzdálenost [m]	Hustota [voz/km]
50	27,78	36,00
90	50,00	20,00
130	72,22	13,85

2.1.1.3 Rychlost

S pojmem rychlost se setkáváme ještě před základním kurzem fyziky a postupně, čím dál jdeme do hloubky daného děje, tím více se to stává zajímavější. Obecně rychlost popisuje ujetou vzdálenost za určitou časovou jednotku:

$$v_p = \frac{r(t_1) - r(t_2)}{t_1 - t_2} = \frac{s}{t} \tag{5}$$

kde: s dráha [m]
$r(t)$ poloha v čase t [m]
t čas [s]

Vztah (5) platí především pro rychlost průměrnou. K okamžité rychlosti se dostaneme pomoci limity, kde čas t_2 bude se blížit k času t_1:

$$v = \lim_{t_1 \to t_2} \frac{r(t_1) - r(t_2)}{t_1 - t_2} = \frac{dr(t)}{dt} = \frac{ds}{dt} \tag{6}$$

Při zkoumání rychlostí v dopravě můžeme zkoumat průměrnou rychlost jednotlivých vozidel projíždějících určitým úsekem a okamžitou rychlost v definovaném místě – to patří k charakteristice mikroskopické. V makroskopickém měřítku zkoumáme především rychlost a cestovní časy na delších úsekách nebo statistickou analýzu mezi skupinami vozidel.

Rychlosti jednotlivých vozidel jsou ovlivněny různými faktory. A to nejenom, například, výkonem vozidla na jednotku hmotnosti, ale i parametry vozovky: stoupaní, poloměry oblouku atd. Všechny dílčí parametry způsobují to, že rychlost každého vozidla je individuální.

Existují dva přístupy, kterými lze měřit průměrnou rychlost. Pokud budeme zaznamenávat okamžité rychlostí jednotlivých vozidel v určitém místě a spočítáme jich průměr – získáme tzv. průměrnou okamžitou rychlost (time-mean speed) a odpovídá jí následující vztah:

$$v_{PO} = \frac{\sum_{i=1}^{n} v_i}{n} \tag{7}$$

kde: v_i rychlosti jednotlivých vozidel $[m/s]$ nebo $[km/h]$
n počet záznamu (součet vozidel) $[-]$

Dalším způsobem, kterým lze spočítat rychlost je zaznamenávání cestovních dob každého vozidla na určitém úseku. Rychlost jednotlivých vozidel získáme pomocí převracené hodnoty příslušné cestovní doby. Pro průměrnou úsekovou rychlost (space-mean speed) platí vztah:

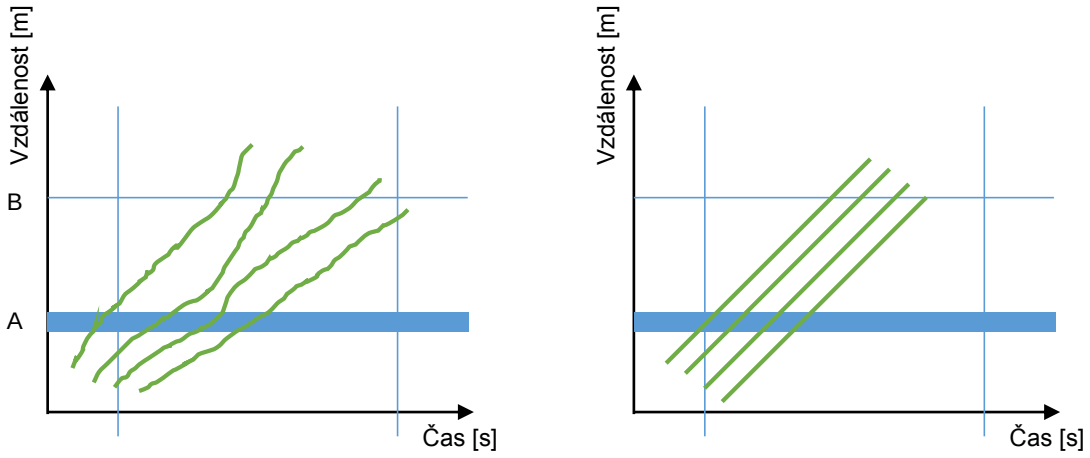
$$v_{PU} = \frac{d}{\sum_{i=1}^{n} \frac{t_i}{n}} \tag{8}$$

kde: d délka úseku $[m]$

t_i cestovní čas vozidla i $[s]$

n počet záznamu (součet vozidel) $[-]$

I kdyby na první pohled mohlo připadat, že výsledné hodnoty budou stejné, ale podobnost může být klamavá. Rozdíl je podoben rozlišení mezi rychlosti úsekovou a rychlosti okamžitou. Okamžitá rychlost může byt ovlivněna mnoha faktory: předjíždění, dokončování manévru, stoupání nebo klesání vozovky a platí pro ni vztah (6) tj. na časoprostorovém diagramu, pokud průjezd bude zaznamenáván v bodě A, bude zobrazena jako tečná ke křivce příslušného vozidla. Ale úseková rychlost mezi body A a B může mít jinou směrnici, než tečná okamžité rychlosti (viz Obrázek 1).



Obrázek 1. Grafy znázorňující rozdíl mezi průměrnou okamžitou rychlostí (vlevo) a průměrnou úsekovou rychlostí (vpravo)

Zdroj: [4]

Samozřejmě existují i vztahy pro převody mezi průměrnou úsekovou a průměrnou okamžitou rychlostmi. Pro poslední platí:

$$v_{PO} = v_{PU} + \frac{\sigma_{v_{PU}}^2}{v_{PU}} \tag{9}$$

kde: $\sigma_{v_{PU}}^2$ rozptyl hodnot průměrné úsekové rychlosti

Obdobný vztah existuje i pro průměrnou úsekovou rychlost:

$$v_{PU} = v_{PO} - \frac{\sigma_{v_{PO}}^2}{v_{PO}} \tag{10}$$

kde: $\sigma_{v_{PO}}^2$ rozptyl hodnot průměrné okamžité rychlosti

Rychlost je jedná z klíčových charakteristik, dle kterých se posuzuje kvalita dopravy, jelikož umožňuje nejenom odhalení kongesci, ale i určování LOS. [5] [6]

Dále nejčastěji používanými pojmy v analýze dopravních proudu jsou:

- Rychlost neovlivněného dopravního proudu – je rychlost, při které vzájemné ovlivňování vozidel je minimalizováno, a chování řidičů není omezeno dopravní situací.
- Provozní rychlost – rychlost, kterou se pohybuje dopravní proud v běžných podmínkách. S klesající hodnotou intenzitou její hodnota se přibližuje rychlosti neovlivněného dopravního proudu.
- Návrhová rychlost – rychlost, pro kterou byla navržená pozemní komunikace.

2.1.2 Měření charakteristik dopravních proudu

Obecně způsobů, kterými lze získávat data je velké množství. Výběr druhu přístupu závisí na mnoha faktorech: jak na výsledných hodnotách, jíž je nutné získat, tak i samozřejmě na dostupných nástrojích. Podle jednoho z klíčových zdrojů [7], odkud jsem čerpal informace, průzkumy lze rozdělit do následujících kategorií:

- Vyhledávání v dokumentech – daný postup je jasný víceméně z názvu. Jedná se o dohledávání potřebných informaci v existujících zdrojích. U nás se může jednat, například, o statistiky z celostátního sčítání dopravy.
- Pozorovací průzkumy – do dané skupiny patří většina průzkumu tykající se dopravy. Pozorovatel zaznamenává výskyt určitého děje a následně zpracovává získána data.
- Průzkumy pro samostatné vyplňování – jedná se především o průzkumy na internetu nebo například o sčítání lidu domů a bytů. Respondent vyplňuje dotazník samostatně bez nutnosti přímého zásahu pozorovatele do procesu vyplňování dotazníku.
- Telefonické průzkumy
- Dotazování v terénu
- Dotazování v domácnostech
- Skupinové průzkumy
- Hloubkové průzkumy

Pro účely získávání charakteristik dopravního proudu, jediný možný přistup je průzkum pozorovací tj. v drtivě většině případů je zapotřebí sledovat pohyb jednotlivých vozidel na komunikaci. Buď to počítat průjezdy vozidel ručně, nebo pomoci detektorů. Ale všechny právě popsané metody umožňují získávání dat bez ovlivnění pohybu dopravního proudu. Je to tzv. *„in-situ“* způsob. Dále lze právě popsaný způsob měření rozdělit do dvou kategorií

- Intruzivní: k intruzivním měřením patří v první řadě detektory, které je zapotřebí zabudovat nebo připevnit k vozovce. Může se jednat, například o:
    - Pneumatické detektory
    - Piezoelektrické detektory
    - Magnetické detektory
    - Indukční smyčky.
- Neintruzivní: jak lze usoudit předchozích definic, neintruzivní měření nezasahují do vozovky a jsou založené na principu vzdáleného pozorování. Patří sem následující způsoby:
    - Ruční sčítání
    - Pasivní a aktivní infračervené detektory
    - Mikrovlnné radary
    - Ultrazvukové a pasivní detektory hluku
    - Videodetekce
    - Kombinovaná měření

Porovnání jednotlivých druhu detektoru a možnosti použití pro měření různých parametru jsou uvedeny v tabulceTabulka 3. [8]

Tabulka 3. Porovnání jednotlivých druhu detektoru a možnosti použití pro měření různých parametru

	Druh detektoru	Intenzita	Rychlost	Klasifikace	Obsazenost	Přítomnost
Intruzivní	**Indukční smyčka**	Ano	Omezeně	Omezeně	Ano	Ano
	Magnetický	Ano	Omezeně	Omezeně	Ano	Ano
	Pneumatický	Ano	Ano	Ano	Ne	Ne
Neintruzivní	**Aktivní IR**	Ano	Ano	Ano	Ne	Ne
	Pasivní IR	Ano	Omezeně	Ano	Ano	Ano
	Mikrovlnný Dopplerův	Ano	Ano	Ano	Ano	Ano
	Ultrazvukový	Ano	Ne	Ne	Ne	Ano
	Pasivní akustický	Ano	Ano	Ano	Ano	Ano
	Videodetekce	Ano	Ano	Ano	Ano	Ano

Ale ne všechny přístupy jsou založeny na principu nezávislého pozorovatele. V poslední době, především kvůli nasazení monitorovacích systému do vozidel, často se používají plovoucí vozidla pro měření charakteristik dopravního proudu.

Výšeuvedené hodnoty lze měřit a používat pro různé účely. Bohužel ne vždy je možné používat nějaké elektronické pomůcky, nástroje a detektory. V některých případech nasazování detektoru může být velmi nákladné především, pokud není nutné měřit hodnoty stále.

Různých druhu dopravních průzkumů je velmi velké množství a v závislosti na požadovaném výsledku lze vymyslet nové druhy. Žádné normy tykající se dané problematiky jsem nenašel, a proto uvedu běžně používané průzkumy:

- Průzkum intenzity dopravního proudu: nejčastěji používaný druh průzkumu, který, jak je patrné z názvu, spočívá v měření počtu vozidel projíždějících určitým profilem komunikace anebo se sčítají intenzity určité křižovatky. Výsledky se používají v podobě vstupních hodnot pro navrhování a plánování pozemních komunikací, řízení uzlu a linie atd.
- Průzkum rychlosti: rychlostní charakteristiky jsou těsně vázané s bezpečnosti na komunikacích a nehodovostí.
- Průzkum cestovních dob: cestovní doby jsou významnou hodnotou pro měření kvality dopravy a často se používá pro poskytování informaci o stavu provozu řidičům.
- Průzkum zdržení: zdržení je podobné cestovní době a mají stejný účel. Navíc zdržení je zapotřebí při plánování řízení dopravy.
- Průzkum hustoty dopravního proudu: jak bylo zmíněno v kapitole 2.1.1.2, hustota se měří obtížně a drtivá většina detektorů zaznamenává vlastní obsazenost a dopočítává hodnotu hustoty.
- Průzkum parkování: zjišťuje se poptávka po parkování v určité lokalitě a to jak z hlediska skladby vozidel, času, účelu atd.
- Průzkum dopravních konfliktů a nehod: jelikož každé nehodě předchází dopravní konflikt, daný druh průzkumu je velmi důležitý pro odhalení problematických úseku a zkoumaní chování řidičů.
- Směrový průzkum: zjišťování zdrojů, cílů a směru pohybu dopravních proudů. Nejčastěji provádí se pomoci dotazování nebo záznamu SPZ.
- Průzkum chodců: lze zjistit chování, rychlost, intenzity.

2.2 Požadavky na software

Jelikož ve své bakalářské práci jsem se zabýval obdobným tématem, požadavky na aplikaci budou vyplývat z mé předchozí práce. V první řadě jsem se soustředil na průzkumy manuální,

jelikož vyžadují maximální přizpůsobivost a pomocí papírových formulářů lze zaznamenávat jakékoliv události. Problém může nastat jedině u získávání dat o rychlosti vozidel, protože bez detektoru jednoduše zjistit daný parametr nelze, protože je zapotřebí zaznamenávat cestovní časy a za velkých intenzit je to velmi obtížné. Taktéž musím poznamenat časovou náročnost ručních měření: vyžadují nejenom stálou přítomnost v měřicím bodě, velkou pozornost při zaznamenávání, ale i přibližně stejné množství času je nutné věnovat přepisu dat.

2.2.1 Požadavky z hlediska uživatele

Celkově software musí usnadňovat samotné průzkumy, jinak nevidím smysl nahrazovat spolehlivé papírové záznamy jakýmkoliv stejně náročným nebo složitějším způsobem. Jednu důležitou věc, kterou může poskytnout použití softwarové aplikace je možnost odstranit časově náročný přepis dat do elektronické podoby. Takže jedním z nejdůležitějších požadavků rozhodně musí být možnost exportu dat do počítače. A to nehledě na druh průzkumu, jelikož považuji daný bod za zásadní.

Výsledky průzkumů jsou z větší části ovlivnění lidskou chybou. Co se týče manuálních záznamu, člověk může špatně rozlišit druh vozidla, čas nebo nechtěně označit průjezd a musí být jemu umožněná následující editace dat. Na papíru je to poměrně jednoduché, vždy je možné chybné údaje zaškrtnout a opravit. Při použití elektronických zařízení s tím mohou vznikat komplikace, ale každopádně považuji možnost editace naměřených dat za nutnou.

Průzkumy s manuálními záznamy není možné provádět pomoci detektorů, a proto pokládám za nutné minimalizovat vliv lidského faktoru. V první řádě je zapotřebí bezpečně ukládat a zálohovat data, aby i při nechtěném vypnutí zařízení naměřené hodnoty byly zálohovaný, a nedošlo ke ztrátě uživatelských dat. Jeden z problému, který pravděpodobně nastane, je výpočetní náročnost stálého zálohování, a proto předpokládám realizaci po určité časově době.

Jedním z dalších zdrojů nepřesnosti, který významně ovlivňuje výsledky je čas. Technické podmínky 189 definuji [2] měření intenzit v časových intervalech ale, například z vlastních zkušenosti vím, jak složité je synchronizovat hodiny mezi jednotlivými stanovišti a kolik problému způsobí jenom půlminutový rozdíl. Současné technologie poskytují velmi velké množství možností pro vyřešení problému jak s nastavením, tak i se synchronizací času. A je to dalším požadavkem na aplikaci.

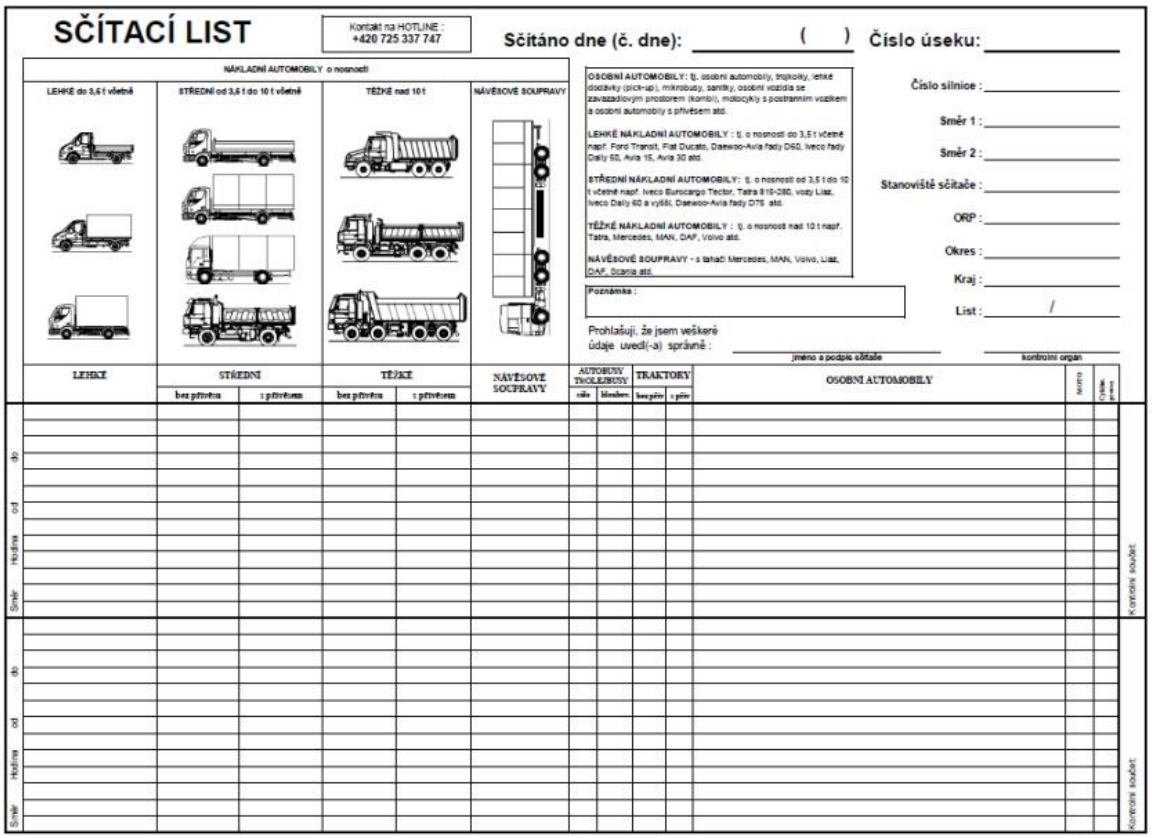
2.2.2 Požadavky z hlediska organizátora

Dále bych soustředil na samotný průběh průzkumů. Pokud hlavním cílem je nahrazení papírových záznamu, v první řádě je nutné se soustředit na odpovídající podobu formulářů a zjistit si co nám poskytují.

Dle doporučení TP189 při sledování intenzit vozidla lze rozdělit [2] do následujících kategorii:

- O osobní automobily – bez přívěsů i s přívěsy, dodávkové automobily,
- M motocykly – jednostopá motorová vozidla bez přívěsů i s přívěsy,
- N nákladní automobily – lehké, střední a těžké nákladní automobily, traktory, speciální nákladní automobily,
- A autobusy – vozidla určená pro přepravu osob a jejich zavazadel, která mají víc než 9 míst (včetně kloubových autobusů a autobusů s přívěsy),
- K nákladní soupravy – přívěsové a návěsové soupravy nákladních vozidel.

Celkově pět kategorií, ale ještě jednou bych upozornil, ž se jedná jenom o doporučení a jak rozdělení, tak, i vzhled formulářů se může lišit dle situace a požadovaných výsledků (například z mého hlediska v rozdělení chybí jednostopá vozidla a cyklisté). Právě je to názorně vidět na jednom z formulářů, který se používal před pěti lety během celostátního sčítání dopravy (viz Obrázek 2).



Obrázek 2. Vzor sčítacího listu

Zdroj: [9]

Na vzoru sčítacího listu lze nalézt celkové 9 kategorií, přičemž některé z nich, jako například střední a těžké nákladní vozidla rozděleny do dílčích podle přítomnosti přívěsu. Podobný

formulář bych požadoval za univerzální, protože pokrývá velkou část vozidel, s výjimkou tramvají). Dále uvedu kategorie z výšeuvedeného formuláře:

- Nákladní automobily o nosnosti:
    - Lehké do 3,5t včetně
    - Střední od 3,5 t do 10 t včetně
        - Bez přívěsu
        - S přívěsem
    - Těžké nad 10 t
        - Bez přívěsu
        - S přívěsem
    - Návěsové soupravy
- Autobusy a trolejbusy
    - Solo
    - Kloubové
- Traktory
    - S přívěsem
    - Bez přívěsu
- Osobní automobily
- Moto
- Cyklisti

Dohromady je to 13 různých druhů, které ale stále nepokrývají celou škálu vozidel vyskytujících se na silnicích. Chybí zde, jak bylo zmíněno, tramvaje, které jsou nezbytnými účastníky silničního provozu ve větších městech. A podle požadavků objednatele průzkumu může být zapotřebí rozlišovat tramvaje dle počtu vagonu nebo přidat i další kategorií události.

Podle všeho lze usoudit, že není možné realizovat jednotný druh formulářů, který by pokrýval všechny potřebné pro zaznamenávání účastníky provozu. Proto není možné jednoznačně stanovit požadavek např. na kategorií vozidel a pro každý průzkum je nutné rozhrání přizpůsobovat a přednastavit několik základních šablon pro nejčastěji probíhající průzkumy.

Lidská povaha je velmi zajímavá a často lidé chtějí podvádět nebo vyhýbat se odpovědnosti a je to vidět nejenom u studentů. Dopravní průzkumy často se probíhají v podobě brigád a je to dalším faktorem, proč lidé mohou být nezodpovědní během zápisu dat: správnost záznamů je těžce kontrolovatelná. V případě že bude existovat možnost jakéhokoliv dohledu nebo kontroly jak průběhu měření, tak i samotných výsledků – určitě podobnou funkci je nutné implementovat. Přičemž je zapotřebí zachovat možnost opravovat chyby v naměřených datech a zároveň zabránit jakémukoliv nahrazování nebo padělání výsledků měření.

2.3 Shrnutí

Po analýze a prozkoumání způsobu měření charakteristik dopravního proudu, různých druhů průzkumů, vzhledu papírových záznamů a potřeb uživatelů, požadavky lze shrnout do následujících bodů:

- aplikace musí nahradit papírové záznamy,
- je nutné vynechat přepis dat ze záznamového přístroje do elektronické podoby,
- uložená data musí mít přehlednou strukturu,
- musí byt realizováno alespoň jedno univerzální rozhraní pro průzkumy intenzit,
- jestli se podaří, implementovat možnost přizpůsobení rozhrání,
- data musí být zabezpečena a zálohovaná,
- uživateli musí byt umožněno opravovat chybné záznamy,
- zaznamenávání polohy stanoviště je vhodné pro kontrolu průzkumu organizátorem.
- synchronizace času umožní dosáhnout přesnějších výsledků.

Dálo by rozepisovat a vymýšlet i další požadavky na software, které musí být realizovány, ale ostatní funkce raději implementují v budoucích verzích aplikace po ověření návrhu.

3 Zabezpečení a ukládaní dat

S průběhem průzkumů je to víceméně jasné a popsáno v předchozích kapitolách. Zbývá vyřešit to, jakým způsobem a v jaké podobě data budou ukládaný. Zaznamenávání do papírových formulářů je jednoduché a spolehlivé až na to, že samotný papír lze ztratit a navíc mohou dělat komplikace přeháňky. Ale ztratit naměřené hodnoty není nejlepší variantou - v takovém případě celý průzkum by byl jenom zbytečnou ztrátou času. V elektronických zařízeních a softwaru celostnost dat je také nutné vyřešit a nejlépe se zálohováním. Z předchozí také kapitoly vyplývá, že je nutné data chránit nejenom proti ztrátě, ale i proti záměrnému poškození uživatelem.

Dále bude provedena analýza různých možnosti ukládání souboru z hlediska jejích formátování, přehlednosti a podporou běžně využívaného softwaru pro zpracování dát z dopravních průzkumů. Následně je vyložena informace o zálohování, ztrátách dat a o tom, jakými způsoby můžeme jím odolávat.

3.1 Druhy souborů

Pokud se jedná o software pro mobilní zařízení, nezbývá nic jiného, než ukládat data do souborů. Nevidím žádný důvod k vytváření vlastních formátů souborů nebo ke komprimaci dat – je to časové náročné a nebude využito v plné míře. Současná zařízeni poskytují dostatek paměti, která umožní neodstraňovat ani zastaralá data.

Mnohem důležitější je volba správné syntaxe pro ukládání nebo struktura souboru. Obsah souboru bude textový, ale jako, například, v případě HTML, data budou uložené v souladu s určitou specifikací. Proto jsem vybral pro daný účel jazyk XML. Dále shrnu nutný minimum, které by mělo vystačit pro porozumění syntaxi uložených souboru.

3.1.1 Základy jazyka XML

XML (eXtensible Markup Language) – rozšiřitelný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C jako doporučený značkovací jazyk. XML byl navržen jako jazyk s jednoduchou formální syntaxí a je vhodný jak pro tvorbu a zpracovaní souborů počítačovými programy, tak i uživatelsky příjemný pro čtení a vytváření dokumentů člověkem. Je rozšiřitelný z hlediska možnosti definování vývojářem vlastní struktury dle požadavků konkrétní aplikace, např. Google používá jazyk XML v souborech typu KML pro ukládání dat na mapových podkladech. A taktéž díky založení na kódování UTF-8 se rozšířil do různých oblastí. [10]

XML dokument má dvě struktury: fyzickou a logickou.

Z hlediska fyzické struktury dokument je sestaven z entit, přičemž každá z nich může odkazovat na další entity. Entitou nazýváme nejmenší část dokumentu. Každá entita má obsah a vlastní název. Obsahem entity jsou symboly a dělí se na dva druhy: znaková data a značky. Ke značkám patří: elementy a atributy.

Logická struktura je sestavená ze dvou částí: prologu a kořenového elementu. Každý XML dokument musí obsahovat kořenový element, který může, ale nemusí mít vložené elementy, znaková data a komentáře. Náplní prologu mohou být deklarace, instrukce a komentáře. Začínat se musí deklaraci XML, ale v některých případech není to povinné.

Je důležité dbát na správné vkládání elementů, protože libovolný element začínající se uvnitř jiného elementu, musí byt uzavřen v tom elementu, ve kterém se objevil. Výjimku tvoří jenom kořenový element.

3.1.1.1 Značkovací symboly

Pro značkování se používají tři základní symboly:

- špičatá závorka otevírací „<“ - používá se pro označování začátku značky,
- špičatá závorka uzavírací „>“ - používá se pro označování konce značky,
- ampersand „&“ – umožňuje nahrazování textu pomoci entit.

Aby bylo možné jednoznačně rozpoznat strukturu dokumentu, tři výšeuvedených symboly nemohou byt použity uvnitř obsahu znakových dat, stejně jako i uvozovky. Místo něj se používají tzv. znakové entity, které nahrazují symboly dle následující tabulky:

Tabulka 4. Přirazení symbolům příslušných entit

Symbol	Entita
<	&lt;
>	&gt;
&	&amp;
‘	&apos;
“	&quot;

3.1.1.2 Prolog

Podle specifikace XML, dokument musí se začínat deklaraci verze jazyka, ve kterém je napsán. Je to dáno odlišnou strukturou pro různé verze jazyka. Pokud deklarace bude chybět, bude považovaná, že dokument je napsaný v jazyce verze 1.0 protože v první verzi deklarace nebyla povinná. Taktéž deklarace může obsahovat informaci o kódování a propojitelnosti s vnějším souborem definice typu dokumentu. Příklad:

```
<?xml version="1.1" encoding="UTF-8" ?>
```

Zdrojový kód 1. Příklad jednoduché definice prologu

nebo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

Zdrojový kód 2. Příklad definice prologu s připojením vnějšího DTD souboru

Ve druhém příkladu atribut „standalone“ odpovídá právě za připojení souboru DTD zevnějšku. Výchozí hodnota je „no“.

Dále následuje deklarace typu dokumentu pomoci instrukce „!DOCTYPE“, která umožnuje pomocí jazyka DTD definovat elementy, atributy, entity aj., které mohou být použity v souboru.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE typ_dokumentu [
  <!ELEMENT pozdrav (#PCDATA)>
]>
<pozdrav>Hello, world!</pozdrav>
```

Zdrojový kód 3. Příklad deklarace typu dokumentu

Anebo pokud budeme chtít načíst DTD dokument z vnějšího souboru, lze druhý až čtvrtý řádek nahradit následujícím obsahem:

```
<!DOCTYPE typ_dokumentu SYSTEM "hello.dtd">
```

Zdrojový kód 4. Příklad deklarace typu souboru s odkazem na soubor DTD obsahující definici struktury XML

kde v souboru „hello.dtd“ bude definovaná značka <pozdrav> stejně, jako kdyby DTD bylo součásti souboru XML.

Komentáře jsou umístěny mezi posloupnostmi symbolu „<!--" a „-->“. Komentář nesmí obsahovat dvě po sobě jdoucí pomlčky.

```
<!-- je to komentář  -->
```

Zdrojový kód 5. Příklad komentáře

3.1.1.3 *Elementy*

Každý neprázdný dokument v jazyce XML musí obsahovat minimálně jeden element. Elementy patří k logické struktuře dokumentu a jsou ohraničeny tagy – počátečním a ukončovacím, které obsahuji stejný název elementu. Pro elementy s prázdným obsahem lze použit tzv. prázdný tag. Příklady použití všech třech typů tagů:

```
<element>
      Element ohraničený počátečním a ukončovacím tágy
</element>
<prazdny_element/>
```

Zdrojový kód 6. Přiklad použití třech typů tagů

kde: <element> - je počáteční tag,

</element> - je ukončovací tag,

<prazdny_element/> - prázdný tag

3.1.1.4 *Kořenový element*

Elementy do sebe mohou být navzájem vnořené, aby odpovídaly struktuře informaci v dokumentu. Jedinou podmínkou je přítomnost tzv. kořenového elementu, který všechny dílčí elementy má zabalené uvnitř. V následujícím přikladu kořenovým elementem je element s názvem „korenovy_element“:

```
<?xml version="1.1" encoding="UTF-8" ?>
<korenovy_element>
  <element1>Obsah prvního elementu</element1>
  <element2>Obsah druhého elementu</element2>
  <element3>Obsah třetího elementu</element3>
  <element4>Obsah čtvrtého elementu</element4>
</korenovy_element>
```

Zdrojový kód 7. Přiklad souboru XML s ukázkou kořenového elementu

3.1.1.5 *Atributy*

K upřesnění významu elementu lze použit atributy, které jsou umístěny uvnitř počátečního tagu. Atribut nemusí být jeden, dovoluje se použití vice atributu pro jeden element. Jenom vždy je zapotřebí uvést atribut i v definici struktury.

```
<element atribut1="hodnota prvního atributu" atribut2="hodnota druhého
atributu">
    Obsah elementu s atributy
</element>
```

Zdrojový kód 8. Přiklad elementu s atributy

3.1.2 Porovnání s ostatními formáty

Existují i další jazyky pro přehledné formátování dat v souborech. Nejčastěji používané jsou: CSV, YAML a JSON. Dále každý z formátu bude stručně popsán a porovnán s formátem XML (viz Zdrojový kód 9). Příklady jsou založeny na reálných hodnotách průzkumu intenzit, kde byly zaznamenány datum, čas a kategorie vozidla.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Průzkum>
    <Vlastnosti>
        <typ>intenzity</typ>
        <Lokalita>
            <souřadnice>50.126787, 14.506857</souřadnice>
            <ulice>Kbelská</ulice>
            <obec>Praha</obec>
        </Lokalita>
        <Začátek>
            <datum>2015-05-16</datum>
            <čas>10:30:00.000</čas>
            <pásmo>UTC+1</pásmo>
        </Začátek>
    </Vlastnosti>
    <Záznamy>
        <událost>
            <Vozidlo>Nákladní střední</Vozidlo>
            <Datum>2015-05-16</Datum>
            <Čas>10:45:6.987</Čas>
        </událost>
        <událost>
            <Vozidlo>Osobní</Vozidlo>
            <Datum>2015-05-16</Datum>
            <Čas>10:45:8.742</Čas>
        </událost>
        <událost>
            <Vozidlo>Osobní</Vozidlo>
            <Datum>2015-05-16</Datum>
            <Čas>10:45:9.132</Čas>
        </událost>
        <událost>
            <Vozidlo>Nákladní souprava</Vozidlo>
            <Datum>2015-05-16</Datum>
            <Čas>10:45:11.190</Čas>
        </událost>
        <událost>
            <Vozidlo>Osobní</Vozidlo>
            <Datum>2015-05-16</Datum>
            <Čas>10:45:14.115</Čas>
        </událost>
    </Záznamy>
</Průzkum>
```

Zdrojový kód 9. Příklad souboru XML obsahující záznamy z průzkumu intenzit

CSV (Comme-Separated Values , hodnoty oddělené čárkami) – textový formát určený pro ukládání tabulkových dat. Každý nový řádek v souboru odpovídá řádku v tabulce. Často pro zajištění kompatibility s některými jazyky, včetně češtiny, které používají čárku jako oddělovač desetinných míst pro zápis čísel, místo čárky se používá středník nebo tabulátor. Jednou z klíčových vlastností souboru CSV je zajištěná podpora tabulkovými procesory (např. nejčastěji používaný Microsoft Excel), jako i v případě s XML. Jak je vidět v přikladu, data jsou přehledně uložena, až na rozdílnou délku jednotlivých řádků. Nevýhodou je chybějící definice formátování a chybějící komentáře. Je nutné pro lepší porozumění struktuře zvlášť definovat

v jakém pořadí jsou data uložená. A v případě dopravního průzkumu je nepřehledné a složité ukládání statických počátečních dat, např. čas zahájení průzkumu, souřadnice, typ průzkumu atd. Výhodou je velmi nízká náročnost na paměťová media bez potřeby komprimace surových dat. [11]

```
Nákladní střední, 2015-05-16,10:45:6.987
Osobní,2015-05-16,10:45:8.742
Osobní,2015-05-16,10:45:9.132
Nákladní souprava,2015-05-16,10:45:11.190
Osobní,2015-05-16,10:45:14.115
```

Zdrojový kód 10. Přiklad souboru CSV

YAML (YAML Ain't Markup Language, YAML není markovácí jazyk) – formát pro zápis sériových dat, který je čitelný jak strojem, tak i člověkem. Hlavní výhodou je minimalizmus, především v porovnání s XML. Zvláštnosti daného jazyka je chybějící podpora tabulátorů, které musí být nahrazeny mezerami. Bohužel nehledě na podporu velkým množstvím programovacích jazyků, není podporován tabulkovými procesory. A to považují za hlavní nevýhodu daného jazyka. Přiklad použití, v porovnání s CSV, je rozšířen o vlastnosti průzkumu: [12]

```yaml
---
Průzkum:
  - Typ: intenzity
  - Lokalita:
     - souřadnice: "50.126787, 14.506857"
     - ulice: Kbelská
     - obec: Praha
  - Začátek:
     - datum: 2016-05-16
     - čas:   10:30:00
     - pásmo: UTC+1
Záznamy:
  - Událost:
      - Vozidlo: Nákladní střední
      - Datum: 2015-05-16
      - Čas: 10:45:6.987
  - Událost:
      - Vozidlo: Osobni
      - Datum: 2015-05-16
      - Čas: 10:45:8.742
  - Událost:
      - Vozidlo: Osobni
      - Datum: 2015-05-16
      - Čas: 10:45:9.132
  - Událost:
      - Vozidlo: Nákladní souprava
      - Datum: 2015-05-16
      - Čas: 10:45:11.190
  - Událost:
      - Vozidlo: Osobni
      - Datum: 2015-05-16
      - Čas: 10:45:14.115
...
```

Zdrojový kód 11. Přiklad zápisu pomocí jazyka YAML

JSON (JavaScript Object Notation, JavaScriptový objektový zápis) je textový formát pro zápis a přenos dat založený na jazyce JavaScript a používaný především tímto jazykem. Stejně jako i výšeuvedené formáty je čitelný jak člověkem, tak i strojem. Pomoci JSON můžeme ukládat pole hodnot, objekty a jednotlivé hodnoty. Je jednoduchý, podporován různými programovacími jazyky a často se používá pro přenos dat mezi web-aplikacemi. Bohužel neumí komentáře, nemůžeme definovat znakovou sadu a v případě znaků zakódovaných do Unicode je nutné používat označování pomoci zpětného lomítka a kódu symbolu. [13]

```
{ "Pr\u016Fzkum" : [ { "Typ" : "intenzity" },
      { "Lokalita" : [ { "sou\u0159adnice" : "50.126787, 14.506857" },
            { "ulice" : "Kbelsk\u00E1" },
            { "obec" : "Praha" }
          ] },
      { "Za\u010D\u00E1tek" : [ { "datum" : "2016-05-16" },
            { "\u010Das" : "10:30:00.000" },
            { "p\u00E1smo" : "UTC+1" }
          ] }
    ],
  "Z\u00E1znamy" : [ { "Ud\u00E1lost" : [ { "Vozidlo" : "N\u00E1kladn\u00ED
st\u0159edn\u00ED" },
            { "Datum" : "2015-05-16" },
            { "\U010Das" : "10:45:06.987" }
          ] },
      { "Ud\u00E1lost" : [ { "Vozidlo" : "Osobn\u00ED" },
            { "Datum" : "2015-05-16" },
            { "\U010Das" : "10:45:08.742" }
          ] },
      { "Ud\u00E1lost" : [ { "Vozidlo" : "Osobn\u00ED" },
            { "Datum" : "2015-05-16" },
            { "\U010Das" : "10:45:09.132" }
          ] },
      { "Ud\u00E1lost" : [ { "Vozidlo" : "N\u00E1kladn\u00ED souprava" },
            { "Datum" : "2015-05-16" },
            { "\U010Das" : "10:45:11.190" }
          ] },
      { "Ud\u00E1lost" : [ { "Vozidlo" : "Osobn\u00ED" },
            { "Datum" : "2015-05-16" },
            { "\U010Das" : "10:45:14.115" }
          ] }
    ]
}
```

Zdrojový kód 12. Přiklad zápisu pomoci jazyka JSON

Ze všech výšeuvedených způsobů dle mého názoru data jsou nejpřehledněji zapsána pomocí jazyku YAML, ale není podporován tabulkovými procesory. JSON má problémy s kódováním, CSV není až tak přizpůsobivý, jak bych očekával. Proto pokládám za jedinou rozumnou variantu použití jazyku XML pro ukládání dat. V následující tabulce je shrnutá celá podkapitola a hodnoty jsou převzaty z výšeuvedených příkladů:

Tabulka 5. Porovnání způsobů formátování

Jazyk/formát	Celkový počet symbolů	Užitečný počet symbolů	Podpora Microsoft Excel
XML	1266	254	Ano
CSV[1]	188	156	Ano
YAML	724	254	Ne
JSON	1530	254	Ne

3.2 Zálohováni

Zálohování samozřejmě patří k nejdůležitějším funkcím aplikace. O naměřená data nechceme přijít a člověk děla chyby mnohem častěji, což může způsobit náhodné mazání nebo ztrátu dat. Hlavním cílem zálohování je vytváření kopie dat s možnosti jejich následného obnovení v případě ztráty původních a to z libovolných důvodů. Pro ukládání záloh běžně jsou používány externí media, ale v poslední době, díky rozšíření cloudových služeb, stále častěji data jsou uloženy na vzdálených serverech.

3.2.1 Přístupy k zálohování

Plná záloha (Full backup) obvykle se tyká celého systému a zálohová všechny soubory v určitých časových intervalech. Ve firemních podmínkách zálohování probíhá o víkendu, aby nezatěžovalo jednotlivé prvky v pracovní dny. V ostatní dny v týdnu zálohy se vytvářejí inkrementálně nebo diferenciálně.

Inkrementální záloha (Incremental backup) provádí kopírování pouze souborů, které byly po poslední plné nebo inkrementální záloze změněny. A proto vždy první inkrementální záloze musí předcházet záloha plná. Samotné zálohování v průměru trvá méně času, protože kopírovaných souborů je málo, ale obnovení je časově náročné kvůli nutnosti obnovit data z poslední plné zálohy a ze všech následujících inkrementálních záloh.

Diferenciální záloha (Differential backup): na rozdíl od inkrementálního zálohování, zálohování diferenciální pokaždé kopíruje všechny změněné soubory po plné záloze a tím zrychluje obnovení dat, ale klade vyšší nároky na velikost záznamových medií.

Nepřetržitá ochrana dat (Continuous data protection) neprobíhá podle plánu, jak tomu je v předchozích případech, ale neprodleně zaznamená každou změnu v systému. Kopírování neprobíhá podle souborů, ale po bytech nebo v blocích dat. [14]

[1] Není zahrnuta informace o typu průzkumu, lokalitě a začátku měření.

3.2.2 Záznamová media

Magnetická páska je jeden z nejstarších a dodnes je velmi rozšířený medium pro ukládání záloh díky příznivému poměru kapacita/cena, ale v současné době rozdíl v ceně mezi pevnými disky a magnetickými pásky je rok od roku menší a menší.

Diskety byly v 80. a 90. letech XX. století nejrozšířenějším osobním přenosným mediem pro ukládání informaci. Nízká kapacita, nespolehlivost disket a cenová dostupnost NAND způsobila zánik daného záznamového media.

Optický disky přišly jako nástupci po disketách. První generace měli kapacitu kolem 0.7 – 0.9 GB, poslední čtvrtá generace nabízí kapacitu až několik terrabajtů. [15] Používá se především pro ukládání multimediálních souborů, ale v posledních letech zájem uživatelů o optická media výrazně klesl.

Pevné disky jsou stále častěji používané pro ukládání záloh. Nabízí možnost bezproblémového nahrazovaní starých záloh novými. Zálohování může probíhat jak lokálně, tak i síťově. Pro zlepšení bezpečnosti dat pevné disky mohou být zapojeny do diskových polí tzv. RAID (Redundant Array of Inexpensive/Independent Disks – vícenásobné diskové pole lacinych/nezávislých disků). [16]

Solid-state media zahrnují jak paměťové karty, USB flash disky ale taktéž SSD disky, které v posledních letech jsou používaný v počítačích. Nabízí vysoké rychlosti čtení a zápisu na úkor ceně za jednotku paměti. Ale díky malým velikostem zastávají velkou část trhu přenosových medii.

Vzdálené služby zažívají ke dnešnímu dni velký pokrok. Poskytovatele internetu nabízí vysokorychlostní připojení za příznivé ceny a je to jeden z důvodu, proč běžné uživatele a správce začínají používat cloudové služby pro ukládání záloh. Hlavní výhodou je malá pravděpodobnost ztráty originálních dát spolu se zálohami kvůli různému fyzickému umístěni. Ale cloudové služby jsou více náchylné vůči hackerským útokům.

3.3 Zabezpečení

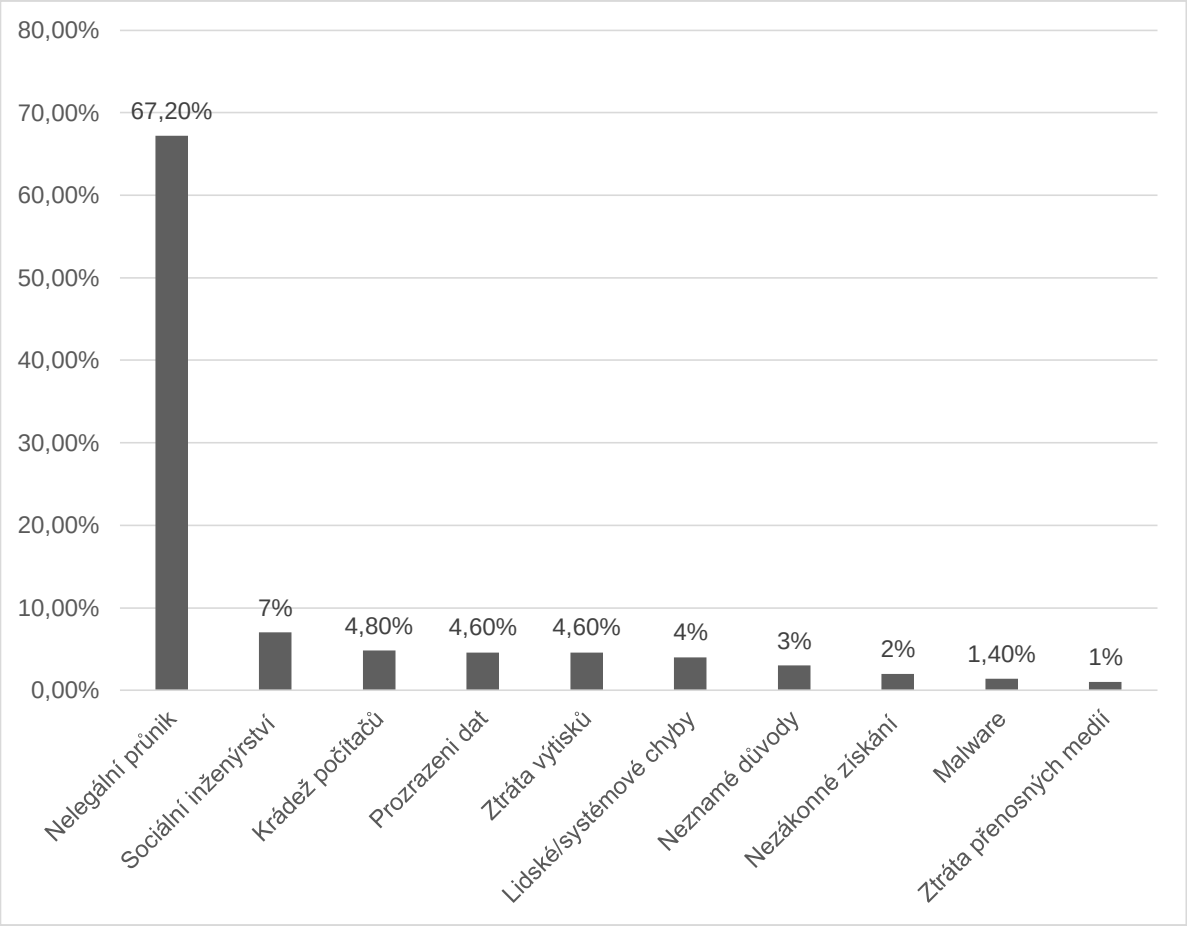
Než se vrhneme na způsoby zabezpečení, důležité je zjistit, proč o data můžeme přijít. Pravděpodobně prevence ztráty může byt méně náročná na realizaci, než implementovat složité zabezpečovací techniky. Například omezení plnohodnotného přístupu k souborům je mnohem jednodušší a snazší než složité šifrování dat.

3.3.1 Příčiny ztráty dat

Samotné příčiny ztráty mohou být různé. Může to být jak chyba softwaru, tak i chyba uživatele. Dále popíšu jednotlivé důvody, kvůli kterým lze o informaci přijít.

- nelegální průnik do systému je dle statistik nejčastější příčinou ztráty dat. Jedná se o neoprávněný přístup ke chráněným informacím nepověřenými osoby. Způsoby mohou být různé: hackeři využívají např. tzv. exploity kterými prolomí ochranu nebo pomoci hrubé sily rozluští přístupová hesla,
- sociální inženýrství se používá pro získání utajené informace nebo přístupu pomoci podvodného jednání,
- krádež počítačů,
- prozrazeni dat v sítí nebo na webu,
- ztráta výtisků dokumentů s citlivými informacemi,
- lidská chyba a chyba systému,
- nezákonné získání dat např. uplacením pracovníku,
- ztráta přenosných medií,
- malware je počítačový software určený pro získání neoprávněného přístupu k informaci v počítači.

Procentuální četnost výšeuvedených příčin je znázorněna na obrázku.



Obrázek 3. Příčiny ztráty dat v roce 2012

Zdroj: [17]

3.3.2 Prevence ztráty dat

Pro účely navrhovaného softwaru postačí se soustředit na prevenci ztráty dat. Ochrana dat proti útokům je zbytečná, jelikož při průzkumu nedisponujeme citlivými a soukromými informacemi. Nejúčinnější způsob ochrany dat proti ztrátám je zálohováni, který pokrývá většinu příčin uvedených v předchozí podkapitole. Přičemž ukládání na vzdálených serverech umožňuje minimalizovat riziko ztráty záloh. Daný přistup je stejně efektivní proti malwaru, ztrátě počítačů nebo přenosových medií s daty a lidským nebo systémovým chybám. I kdyby ztráty byly způsobeny uživatelem, který data náhodou odstranil nebo kvůli poruchám výpočetní techniky, pravděpodobnost ztráty původních dat a zálohovaných je malá.

Antivirový software a firewall brány spolu se správným nastavením přístupových prav, umožňuje data ochránit vůči neoprávněnému přístupu. Antivirusem lze detekovat víry a škodlivé počítačové programy. Firewall může zabránit nelegálnímu průniku do systému stejně, jako nastavení přístupových prav.

Lidskému faktoru a jím způsobeným chybám lze čelit správným vyškolením. Zkušené uživatele jsou méně náchylné vůči fishingu nebo sociálnímu inženýrství. Ale daný způsob nezaručuje stejnou spolehlivost, jako, např. omezení práv nebo přístupu.

4 Analýza a volba vhodného nástroje pro tvorbu softwaru

Poslední dobou přenosná výpočetní zařízení stále více pronikají do různých sfér našeho života. Chytré mobily vytlačily klasické mobily díky široké škále nabízených služeb a aplikací. Ale otázkou je, kolik uživatelů skutečně využívají výpočetní výkon zařízení, co mají v kapse. Příznivé ceny spolu s nabízenými možnostmi zapříčinily využití chytrých mobilu, například i pro firemní účely nebo ve sféře prodeje a služeb. V restauračních podnicích se lze setkat nejenom se speciálně vyvinutými POS terminály, ale i s běžně používanými mobily nebo tablety, přičemž někde chytrá mobilní zařízení nahrazují dokonce i vytištěné jídelní lístky.

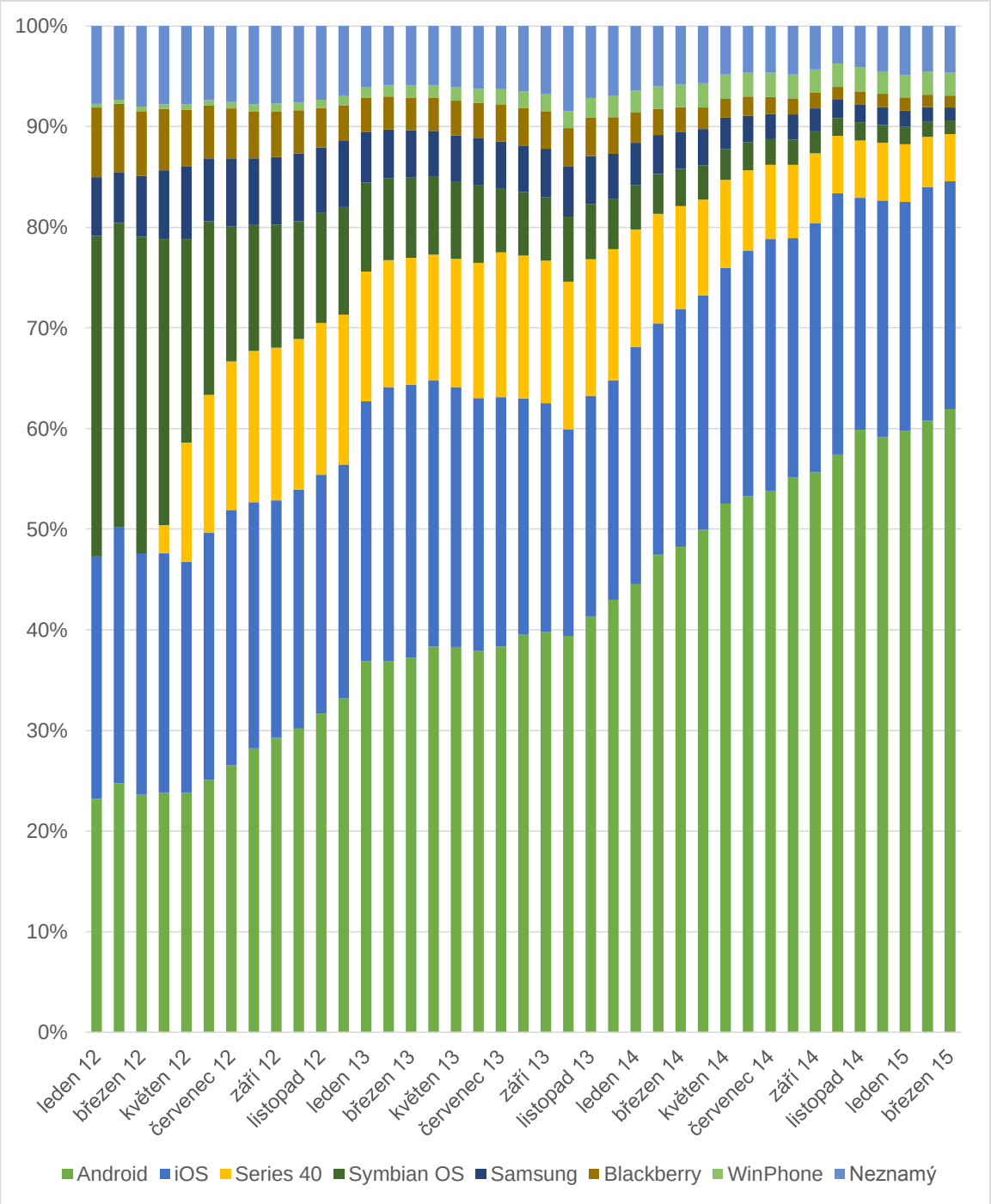
Další sférou, ve které použití mobilních zařízení pro firemní účely, se stálo běžnou záležitosti, jsou taxi služby. [18] Aplikace propojují zařízení zákazníka se zařízením řidiče a vynechávají mezikrok v podobě dispečinku. Cesta, kterou volí poskytovatele taxi služeb, upřednostňovává vývoj aplikace pro chytré telefony před vývojem vlastní OBU jednotky. Tvorba softwaru je méně náročné v neposlední řadě i z finančního hlediska.

A právě proto jsem se věnoval návrhu softwaru pro chytré telefony. Trh mobilních zařízení v současné době je velmi nasycený nabídkami. Desítky výrobců nabízí stovky modelů, které mohou splnit jakékoliv přání zákazníka: nárazuvzdorné, voděodolné, luxusní nebo cenově dostupná zařízení. Ale podstatná je jedná věc: množství operačních systémů, které jsou nainstalovány v chytrých mobilech a tabletech, je zastoupené jenom několika klíčovými společnostmi. Jedná se především o Google, Apple a Microsoft se svými produkty. Ale existují, samozřejmě, i další korporace, které vyvíjí operační systémy pro chytrá mobilní zařízení.

4.1 Analýza operačních systému v segmentu mobilních zařízení

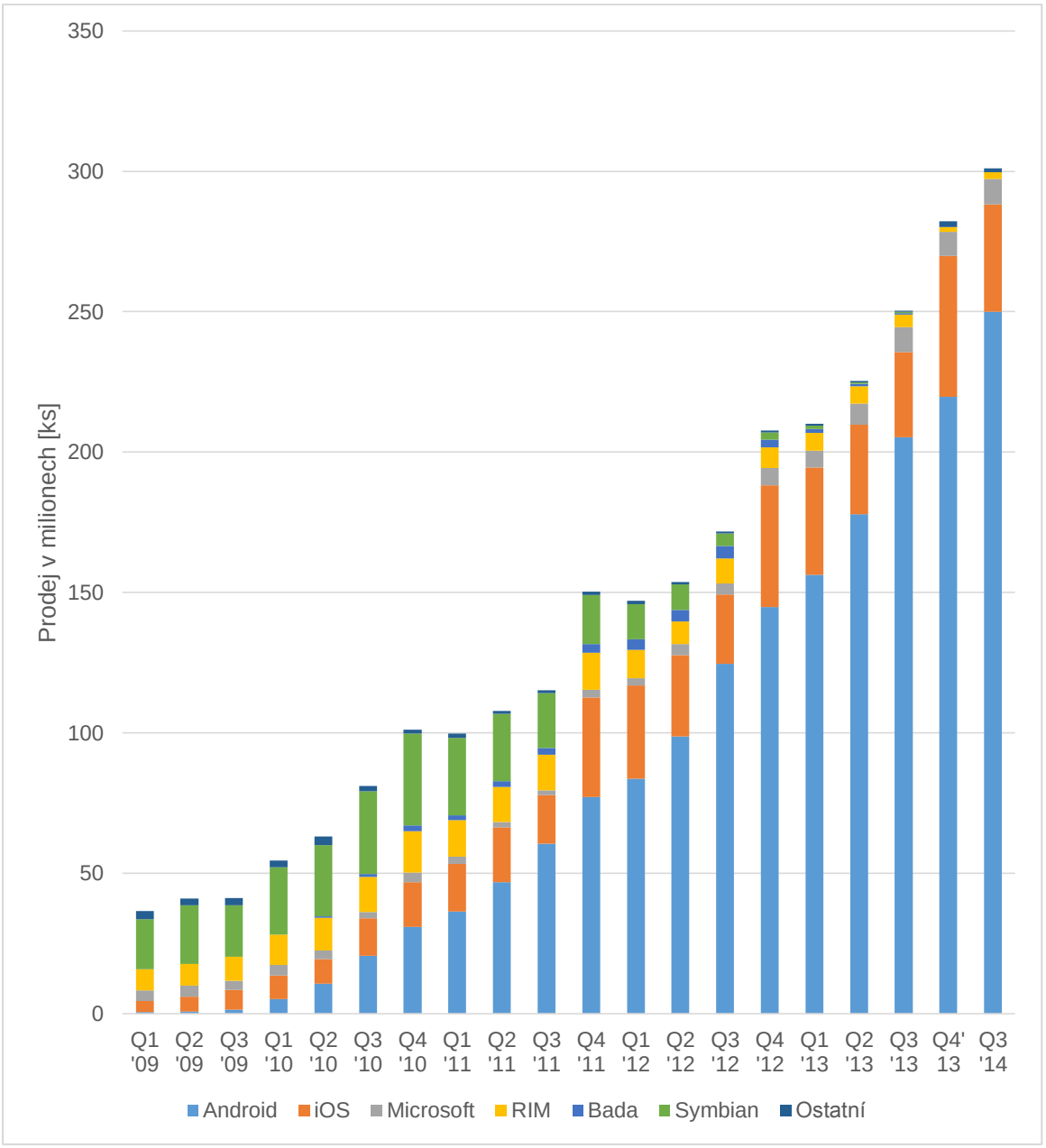
Zastoupení jednotlivých operačních systému na trhu je velmi nerovnoměrně. Některé výrobce jsou soustředěny na vývoje operačního systému a poskytují ho jako službu, jiné se zabývají prodejem přístrojů a nemají zájem o rozšíření jimi dodávaného softwaru na zařízení jiných výrobců. Proto je potřeba zanalyzovat trh mobilní elektroniky z hlediska podílu jednotlivých operačních systémů a zaměřit se především na nejrozšířenější.

Musel jsem zapátrat po statistikách a podařilo se mi nalézt následující grafy. První graf byl získán z dat o počtu zařízení, která brouzdají sítí internet.



Obrázek 4. Zastoupení operačních systémů v segmentu mobilních zařízení

Zdroj: [19]



Obrázek 5. Celosvětový prodej chytrých mobilu s rozdělením podle jednotlivých operačních systémů

Zdroj: [20]

Z grafu lze posoudit, že největší podíl na trhu má operační systém Android. A to jak v počtu prodaných zařízení koncovým zákazníkům, tak i dle podílu při surfování na internetu. Je také vidět stoupající tendence zastoupení Androidu na trhu. Jeho velkou výhodou je, že Google poskytuje komunitě vývojářů možnost úpravy operačního systému dle svých požadavků, což nabízí např., přistup k hardwaru. Proto zaměřil jsem se právě na chytrá mobilní zařízení s operačním systémem Android. Ale podíváme se i na další.

Sekundárním hráčem je společnost Apple se svým systémem iOS. Obdivuhodné je, že mají mnohem menší nabídku zařízení, ale stejně zastávají velkou část trhu. Vývoj pro iOS není tak atraktivní, jako pro Andoid, především z důvodů vyšších pořizovacích nákladů a taktéž kvůli malé nabídce zařízení.

Zastoupení ostatními zařízení pokládám za nevýznamné. A například, co se týče zařízení Blackberry nebo Amazon, které mají vlastní operační systémy založené na Androidu, a umožňující bez problému aplikace vyvinuté pro Android spouštět. Podle aktuálních informací Windows Phone také bude schopen spouštět Android aplikace. [21]

4.1.1 Android

Operační systém Android, který od roku 2005 patří společnosti Google, je platforma pro chytré mobily, tablety, čtečky knih, multimédia přehrávače, televize, nositelnou elektroniku a počítače. Byl vyvinut primárně pro dotyková zařízení, ale ostatní zařízení jsou podporovány za použití zvláštních uživatelských rozhrání. Systém je založen na jádru Linux a vlastní realizaci virtuálního strojů Java společnosti Google. Operační systém má otevřený zdrojový kód spadající pod licenci GNU GPL v2, která umožňuje výrobcům a vývojářům za dodržení licenčních podmínek snadné použití a dolaďování OS dle svých představ. Ve skutečnosti zařízení dodávaná výrobci skrývají uvnitř spojení otevřeného softwaru s proprietárním včetně i aplikaci vyvíjených společnosti Google.

Software pro operační systém Android lze vyvíjet v jazyce Java za použití knihoven prostřednictvím tzv. Android Application Framework. Navíc existuje možnost díky Android Native Development Kit importovat knihovny a časti aplikaci vyvinuté v jiných programovacích jazykách.

4.1.2 iOS

Operační systém iOS, do roku 2010 taktéž známá jako iPhone OS, je operační systém pro chytré mobily, tablety a nositelnou elektroniku, která patří a je vyvíjená americkou společnosti Apple. Na rozdíl od konkurentů: Android a Windows Phone, je určen pouze pro vlastní výrobky, jako jsou iPhone, iPod, iPad atd.

Operační systém iOS byl vyvinut na základě stolního operačního systému OS X, který používá společnost v počítačích vlastní výroby. Používá stejné hybridní jádro XNU, které patří do rodiny Unix podobných systémů. Nehledě na to, že jádro má otevřený zdrojový kód s licenci APSL 2.0, samotný operační systém je poměrně uzavřený. Uživateli má omezené možnosti instalace vlastních aplikaci – jediná legální možnost je přes oficiální správce balíčků App Store.

Vyvíjet aplikace lze v jazyce C nebo Objective-C. Apple dlouhou dobu neumožňoval používat vlastní vývojová prostředí a vše bylo omezeno jenom použitím aplikaci XCode, která je i do

dnes k dispozici jenom pro počítače s operačním systémem OS X. V současné době jsou i alternativy umožňující, mimo jiné, vývoj multiplatformního softwaru. Jak jsem již zmínil, aplikace lze instalovat jenom z oficiálního App Store, ve kterém procedura umístění není jednoduchá.

4.1.3 Windows

Společnost Microsoft lze pokládat za průkopníka mobilních operačních systémů. První generace, taktéž známá jako Windows Mobile byla vyvinutá pro vlastní platformu kapesních počítačů Pocket PC ještě před 15 lety v roce 2000. Poslední verze byla zveřejněna 2. února 2010.

Její nástupcem stal operační systém Windows Phone. První verze s číslem sedm v názvu byla odhalená 15. února na Mobile World Congress 2010 a stala dostupná konečným zákazníkům 21 října téhož roku. Aktuální verze nese označení Windows Phone 8.1. V testovací fázi se nachází další verze, která podlé různých zdrojů bude se nazývat Windows 10 Mobile nebo Windows Phone 10. [21]

Windows Phone je proprietárním softwarem s vlastní licencí Microsoft EULA. První verzi zdědili jádro Windows CE od Poket PC. Novější verze OS obsahují přizpůsobené jádro Windows NT. Vývoj aplikaci nativně je podporován v jazyce C++ nebo v případě starších verzi OS: C# a Visual Basic .NET.

Bohužel ani do dnes společnosti Microsft se nedaří úspěšně konkurovat s ostatními obry na trhu mobilních operačních systémů a podíl Windows na přenosných zařízeních, na rozdíl od stolní verze systému se stejným názvem, je neuvěřitelně malý.

4.1.4 Ostatní mobilní operační systémy

Samozřejmě trh mobilních operačních systémů není omezen jenom třemi výšeuvedenými platformy, ale soustředěn kolem něj. Existují další OS, které jsou vyvíjený buď entuziasty a komunitou nebo vývoj setrvává bývalými lídry na trhu. Nebudeme zabíhat do podrobností jednotlivých OS, jelikož jejích podíl je tak nevýznamný, v porovnání s Androidem nebo iOS, a proto uvedu jenom jejich seznam:

- BlackBerry OS: operační systém vyvíjený společnosti Research In Motion pro vlastní zařízení (tablety a chytré mobily) a je zaměřená především na firemní klientelu díky nabízeným B2B funkcím. [22]
- Bada: vyvíjená společností Samsung Electronics mobilní platforma s proprietárním zdrojovým kódem pro mobilní telefony s dotykovými obrazovkami, chytré televize a chytré hodinky. K současnému datu projekt je ukončen. [23]

- Tizen je nástupcem platformy Bada, kterou vyvíjí Intel a Samsung jako členy společnosti Technical Steerigng Group, a taktéž je podporována Linux Foundation spolu s Tizen Association. Operační systém založen na jádru Linux a určen pro různá výpočetní a multimediální zařízení včetně mobilu, televizi, tabletu atd. na architektuře x86 nebo ARM. [24]
- Firefox OS je operační systém s otevřeným zdrojovým kódem, jehož klíčovou vlastnosti je, že celé uživatelské rozhrání představuje webovou aplikaci, která zobrazuje a spouští ostatní programy napsané pomocí HTML, CSS a JavaScriptu. Jejích komunikace s hardwarem je realizovaná přes vlastní API. Vývojem se zabývá Mozilla Foundation na základě renderovacího jádra Gecko [25]
- Ubnuntu Touch je produktem společnosti Canonical Ltd. s otevřeným zdrojovým kódem a patřící do rodiny Unix-like operačních systémů. Jednou z klíčových vlastností je podpora zařízení s původně předinstalovaným Androidem. [26]

4.1.5 Shrnutí

Volba operačního systému pro software není tak složitá, jak jsem původně očekával. Soustřeďovat se na softwaru pro málo rozšířené operační systémy není nejlepší nápad. Proto bylo zapotřebí vybírat mezi operačními systémy Android, iOS a Windows Phone.

Windows Phone, v porovnání s ostatními, ani se řadově nepřiblížil v obsazení pozic na trhu k výrobkům společností Google a Apple. Navíc má problémy se zpětnou kompatibilitou [27] a není zaručeno, že software bude pracovat i v prostředí novějších nebo starších verzi operačního systému Windows Phone.

Nabídka výrobku s iOS je velmi chudá a z mého hlediska je cenově nepřijatelná pro účely dopravních průzkumů. Komplikace s instalací aplikaci je rozhodující pro volbu jiné platformy.

Operační systém Android pro účely dané diplomové práci je vice než postačující. Široké cenové rozmezí výrobků umožní zvolit zařízení, které by nejlépe odpovídalo požadavkům na měření. Jednoduchá instalace a možnost přístupu k systémovým nastavením ve výsledku dělá z Andoidu velmi atraktivní platformu pro vývoj jak běžných aplikaci, tak i specifického softwaru.

4.2 Analýza vývojových prostředí a nástrojů

V předchozí kapitole byly popsány jednotlivé mobilní operační systémy a rozhodl jsem, že nejlepší variantou pro vývoj softwaru pro podporu dopravních průzkumu je OS Android. Dále zbývá zvolit vývojové prostředí. Obvykle vývoj aplikaci probíhá v programovacím jazyce Java s použitím Android Software Developnet Kit, ale není to jediná varianta. Ostatní společnosti nabízí vlastní nástroje pro vývoj podporující ostatní programovací jazyky a umožňují

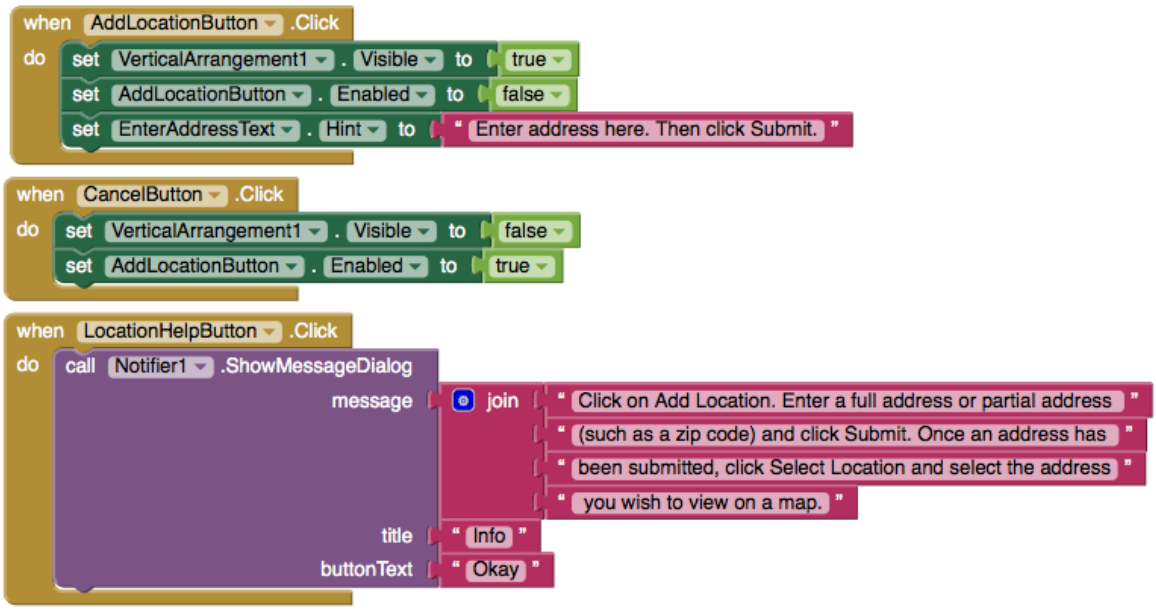
jednoduchou migraci aplikace na jiné platformy. Podíváme se na jednotlivé možnosti podrobněji.

4.2.1 Android studio

Android studio je oficiální vývojové prostředí založené na IntelliJ IDEA a určeno speciálně pro vývoj softwaru pro operační systém Android. Vývojem samotného Android Studio se zabývá společnost Google. Je dostupné zdarma dle licence Apache License 2.0 pro operační systémy Windows, MacOS a Linux. IDE nabízí WYSIWYG editor, živý render aplikace, podepisování aplikace, průvodce založené na šablonách, vestavenou podporu Google Cloud Platform. Pokud je nutné vyvíjet aplikace pouze pro Android – je to nejlepší volba, jak z hlediska obsáhle dokumentace, tak i kvůli komunitní podpoře. Má vestavenou podporu emulátoru z oficiálního balíčku Android SDK. Podporuje nejenom chytré mobily a tablety, ale taktéž i chytré hodinky, brýle Google Glass a Android TV. Nabízí jednoduchou správu překladů aplikace díky integrovanému správci Translation Editor. [28]

4.2.2 App Inventor for Android

App Invertor je velmi zajimavým projektem, který původně byl vyvinut společnosti Google, ale v současné době patří Massachusettskému technologickému institutu. Je podobny programovacímu jazyku Scratch tj. umožňuje lidem s jakýmikoliv programovacími zkušenostmi vyvíjet software pomocí drag-and-drop interakce s vizuálními objekty, v podobě jednotlivých procedur, funkci operátorů atd. Velkou výhodou je práce ve webovém prohlížeči bez nutnosti instalovat na počítač jakýkoliv software. Umožňují přístup k senzorům, funkcím volání, SMS zprav, kontaktům, uložišti a jiným základním prvkům zařízení. [29]



Obrázek 6. Přiklad zdrojového kódu ve vývojovém prostředí App Invertor for Android

Zdroj: [29]

Jedná se o skvělou variantu pro ty, kdo nemá zkušenosti nebo hluboké znalosti v oblasti programování. Pokud požadavky na software nejsou zvlášť specifické a není potřeba zabývat se vývojem profesionálně, lze díky App Invertoru ušetřit čas a během několika minut vytvořit jednoduchou aplikaci a mít jí připravenou k instalaci na zařízení.

4.2.3 Corona

Corona SDK je proprietární baliček pro vývoj software, který patří společnosti Corona Labs Inc. a umožňuje vyvíjet aplikace pro mobilní zařízení s operačními systémy Android, iOS Windows Phone a Kindle v jazyce Lua. Vývojové prostředí může být spuštěno na stolních počitačích s OS Windows nebo Mac OS X. [30]

Corona je určena především pro vývoj dvoudimenzionálních grafických aplikaci. K základním vlastnostem patří podpora audia a grafiky, šifrování, síťového rozhraní, přistup k senzorům a uživatelskému zadávaní dat.

Obrázek 7. Příklad aplikace napsané v jazyce LUA ve vývojovém prostředí Corona

Zdroj: [30]

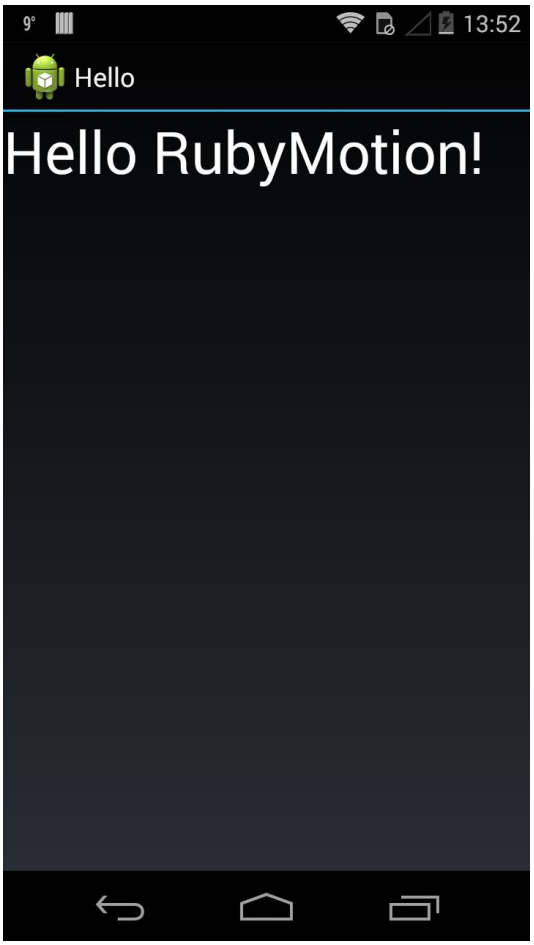
Základní verze softwaru je k dispozici zdarma a měla by postačit pro nenáročné uživatele. Doplňkové funkce jsou zpoplatněny měsíčným poplatkem ve velikosti 79 USD nebo 199 USD.

4.2.4 Kivy

Knihovna s otevřeným zdrojovým kódem, kterou vyvíjí Kivy organization umožňuje návrh a tvorbu aplikaci nejenom pro mobilní zařízení, ale i pro dotyková zařízení s operačními systémy Linux, Windows a OS X. Spadá pod podmínky licence MIT a je distribuovaná zdarma. Vývoj probíhá v jazyce Python. U zařízení se systémem Android je podporován přístup k senzorům, notifikacím, kameře, SMS a e-mailovým zprávám.

4.2.5 RubyMotion

RubyMotion je proprietárním produktem společnosti HipByte, který původně byl určen pro vývoj softwaru pro zřízení s iOS v jazyce Ruby. V prosinci roku 2014 byla anoncovaná podpora operačního systému Android. Bohužel k systémovým požadavkům patří operační systém OS X 10.8.4 nebo vyšší, který může být nainstalován pouze na počítače společnosti Apple a proto vývoj aplikace může vyvolat vysoké náklady při pořizování hardwaru pro vývoj. [31]




```
class MainActivity < Android::App::Activity
  def onCreate(savedInstanceState)
    super

    @text =
Android::Widget::TextView.new(self)
    @text.text = 'Hello RubyMotion!'
    @text.textColor =
Android::Graphics::Color::WHITE
    @text.textSize = 40.0
    self.contentView = @text
  end

  def dispatchTouchEvent(event)
    @counter ||= 0
    case event.action
      when
Android::View::MotionEvent::ACTION_UP
        @counter += 1
        @text.text = "Touched #{@counter}
times!"
        @text.backgroundColor =
Android::Graphics::Color::BLACK
      when
Android::View::MotionEvent::ACTION_MOVE
        @text.text = "ZOMG!"
        @text.backgroundColor =
Android::Graphics::Color.rgb(rand(255),
rand(255), rand(255))
      end
    true
  end
end
```

Obrázek 8. Příklad aplikace vytvořené pomocí softwaru RubyMotion

Zdroj: [31]

4.2.6 Xamarin

V květnu roku 2011 v San Francisku vznikla společnost Xamarin, která vyvinula otevřenou specifikaci Common Language Infrastructure a ke dnešnímu datu nabízí celý baliček nástrojů pro vývoj mobilních multiplatformních aplikaci pro zařízení s operačními systémy Android, iOS a Windows Phone v jazyce C#. Společně s vlastním IDE jsou nabízený nástroje pro testování, ladění, agregaci chybových hlášení analýzu stability aj. Software patří k proprietárním a základní verze, která je poskytovaná zdarma, má velmi omezené možností vývoje, ale postačující pro jednoduché aplikace o maximální velikosti 128 kb. V ostatních případech jsou k dispozici různé plány s měsíčním poplatkem 25 – 158 USD:

4.2.7 Porovnání a volba vývojových nástrojů

Popsal jsem jenom zlomek ze všech nabízených nástrojů, IDE a SDK pro vývoj mobilních aplikaci. Ve skutečnosti je jich mnohem více a jenom při prvním a rychlém zkoumání podařilo

se mi najít více než padesát různých softwaru pro vývoj. Dále uvedu porovnávací tabulku nástrojů, zmíněných v předchozí podkapitole a podle ní bude zvoleno vývojové prostředí pro následující vývoj.

Tabulka 6. Porovnání vývojových prostředí

Platforma	Programovací jazyk	Podporované OS	Požadavky	Cena
Android studio	Primarně Java, některé části mohou být napsáne v C a C++	Android	Windows, Linux a OS X	Zdarma
App Invertor for Android	Vlastní vizuální jazyk	Android	Libovolný OS s webovým prohlížecím	Zdarma
Corona	Lua	iOS, Android, Windows Phone, Kindle	Windows a OS X	0 – 199 USD měsíčně
Kivy	Python	Linux, Windows, OS X, Android a iOS	Linux, Windows, Android a iOS	Zdarma
RubyMotion	Ruby	iOS a Android	OS X	Od 16 USD měsičně
Xamarin	C#	iOS, Android, Windows a OS X	Windows a OS X	0 – 158 USD měsičně

Z tabulky je patrné, že nehledě na překrývající se možností, které nabízí jednotliéi nástroje, požadavky a ceny se liší. Tak v první řadě jsem musel škrtnout RubyMotion z toho důvodu, že nedisponuji zařízením, přesněji řečeno počítačem, s operačním systémem OS X a jeho pořizování pro účely diplomové práci nebylo by vhodné především nákladově.

Dále jsem odmítl volbu jakéhokoliv softwaru, který by v budoucnu požadoval rozšíření na placenou verzi. Zaprvé jak z hlediska toho, že během vývoje může vyskytnout potřebnost používat některé zpoplatněné funkce, tak i kvůli tomu, že navrhovaný software

mám v plánu rozvíjet i mimo danou diplomovou prací. Právě proto nejsem schopen garantovat, že vystačím pouze se základními verzemi vývojových prostředí.

Zbyly jenom originální Android studio a App Invertor for Android s Kivy od vývojářů třetích stran. Poslední se mi nezdá nejlepší volbou, jelikož s programovacím jazykem Python mám nejméně zkušeností a z dostupných zdrojů [32] je patrné, že knihovna Kivy je zaměřena ve větší míře pro stolní operační systémy nebo pro specifická zařízení s vlastně navrženými způsoby HMI.

App Invertor for Android byl by velmi vhodnou volbou, pokud bych vývoj softwaru zanechal pouze ve stadiu diplomové práce. S podobnými důvody, které jsou popsané o dva odstavce výše, vidím komplikace v budoucnu, pokud bude zapotřebí software rozšířit o nové nebo specifické funkce. Taktéž mě zarazily omezené možnosti úpravy grafického rozhraní např. umístění a velikost jednotlivých prvku View je omezeno pouze třemi možnostmi: automatické přizpůsobování, zdědění po rodičovském prvku a pevné zadávaní v pixelech. Ale operační systém je schopen umístit prvky např. vzhledem k určité hranici obrazovky absolutně nebo relativně a v různých jednotkách. Bohužel může to být zásadním problémem, protože grafické rozhrání pro dopravní průzkumy pravděpodobně bude vyžadovat specifické umístění prvků na obrazovce, a rozhraní by se mělo přizpůsobovávat pro různá zařízení s odlišnými úhlopříčkami displejů.

Jedinou vyhovující variantou je použití oficiálního vývojového prostředí od společnosti Google, jehož výhodami je nativní podpora všech možností operačního systému, absence jakýchkoliv poplatků za použití, rozsáhla dokumentace a komunitní podpora. Nevýhody jsou jenom v podobě časové náročnosti při vývojů, v porovnání s App Invertorem a zaměřenosti pouze na jeden operační systém.

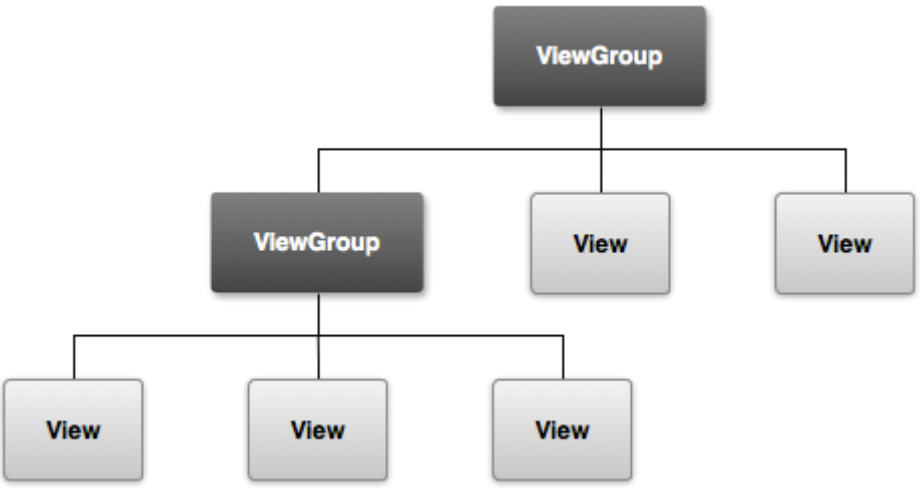
5 Návrh a vývoj softwaru

Návrh aplikace bude vycházet především z mé bakalářské práce, odkud jsem čerpal poznatky o rozložení vzhledu. Jako vývojové prostředí jsem zvolil Android studio. Na začátku je nutné vybrat verzi API, která odpovídá verzi operačního systému. Android studio při založení nového projektu navrhuje verzi v závislosti na statistikách, kterými disponuje Google, a poskytuje data o tom, jaký procent zařízení bude schopen spouštět aplikaci.

Ale na začátku bych stručně pověděl o jednotlivých prvcích, ze kterých aplikace jsou sestaveny. Názvy ponechám nepřeložené ve zdrojovém jazyce, který je uveden v dokumentaci, a který se používá v IDE a SDK.

Activity – je v podstatě jedna obrazovka aplikace s umístěnými na ní jednotlivými prvky View, pomoci kterých probíhá interakce s uživatelem. Obvykle jedno okno Activity překrývá celou obrazovku zařízení. Aplikace může obsahovat více komponent Activity. Při spouštění, například, uživatel uvidí jednu Activity, další Activity se bude zobrazovat na obrazovce s nastavením aplikace atd. Každá Activity má svůj životní cyklus: spouštění, pozastavení, zastavení a pokračovaní, ukončení aj.

Obecně jakýkoliv prvek uživatelského rozhrání na obrazovce nese název View. Může se jednat například o tlačítka, textová pole, obrázky, tabulky apod.. Umístěny musí být hierarchicky a v tom pomáhá prvek ViewGroup, který umožňuje sdružení do skupin, jak je zobrazeno na obr. XX. Deklarovat jednotlivé prvky lze buď v zdrojovém kódu aplikace anebo existuje i jednodušší varianta pomocí úkladní rozhrání do XML souboru.



Obrázek 9. Ukázka seskupení prvků View

Zdroj: [28]

Pokud aplikace nebude obsahovat uživatelské rozhraní, podobný komponent aplikace se nazývá Service. Umožňuje aplikaci provádět dlouhodobou aktivitu na pozadí nehledě na to, kterou aplikaci uživatel používá v současný okamžik.

Content provider: jak je patrné z názvu, jedná se o prvek poskytovatele obsahu, který řídí přístup ke strukturovaným datům. Poskytuje mechanizmus pro definování zabezpečení dat a taktéž nabízí standardizované rozhraní pro přenos dat mezi jednotlivými procesy.

Broadcat receiver je prvkem, který přijímá zprávy, které jsou rozesílaný v celém systému. Zprávy mohou být rozesílány operačním systémem za různých okolností: při nízkém napětí baterie, vypnutí obrazovky aj. Taktéž aplikace mohou rozesílat svoje vlastní zprávy, aby zajistili včasnou reakci ostatních aplikaci na určitou operaci. Zkratka na cokoliv, záleží na vývojářovi.

Po krátkém úvodu, doufám, že čtenářovi by měl být následující text mnohem srozumitelnější.

5.1 Uživatelské rozhrání

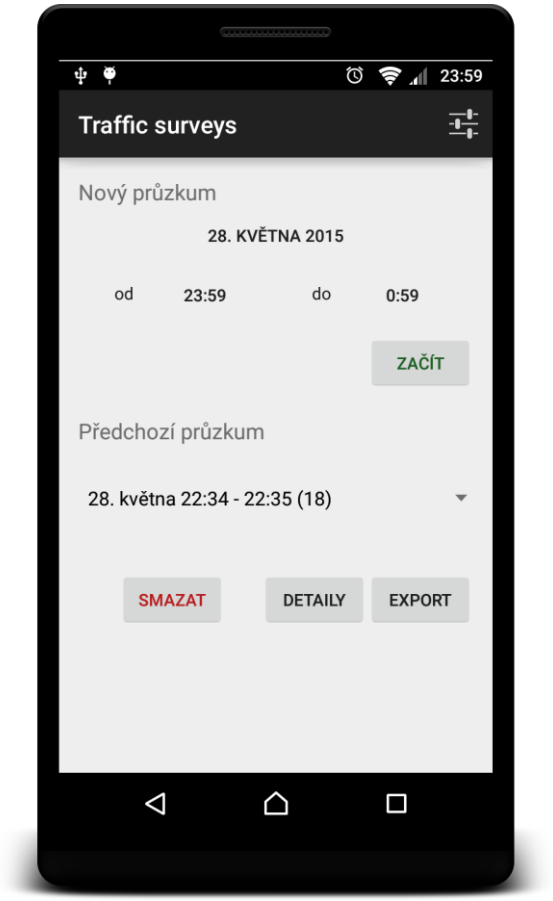
Dalším krokem po instalaci IDE je návrh uživatelského rozhrání. Je nutné definovat počet Activity, jejích vzhled a rozložení prvků, určit jakým způsobem uživatel bude přecházet od jedné obrazovky ke druhé. Vycházel jsem z požadavku na software: zaměření na průzkumy intenzit. Proto bylo zvoleno rozhraní ze třech obrazovek:

- úvodní obrazovka – především musí poskytovat přístup k následujícím obrazovkám,
- obrazovka pro průzkumy – slouží pro samotné zaznamenávání dat a bude zobrazována uživateli během měření,
- obrazovka s předvolbami – poskytuje rozhrání pro nastavení aplikace.

V dalších kapitolách následuje detailnější popis zmíněných obrazovek včetně rozložení, prvků a funkcí.

5.1.1 Úvodní obrazovka

První obrazovka, která se objeví před uživatelem a nese název MainActivity, nesmí být vytížená různými prvky, ale zároveň musí poskytovat přístup k dalším Activity. V našem případě se musíme dostat buď na obrazovku s měřením, nebo na obrazovku s předvolbami. Na Obrázek 10 je znázorněn snímek úvodní obrazovky.



Obrázek 10. Rozložení úvodní obrazovky

Zdroj: autor

Přistup k obrazovce s nastaveními je realizován pomocí tlačítka, umístěného v horní nabídce tzv. Action Bar. Ve výchozím nastavení je schováno do rozevírací nabídky s výhledem, že menu bude obsahovat více jednotlivých položek. Ale nevidím důvod k existenci většího množství tlačítek pro nastavení nebo dalších nabídek, jelikož aplikace je poměrně jednoduchá. Proto jsem rozevírací menu nahradil jednou ikonou, neboť grafické ikony jsou více uživatelsky přívětivé, a pokud pro ni nezbyde místo (spíše výhled do budoucna v případě přidaní dalších ikon do Action Baru), schová se, jak je tomu ve výchozím nastavení. Zdrojový kód v XML formátu, který určuje rozhrání horní nabídky je následující:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
tools:context=".MainActivity">
    <item
        android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:icon="@drawable/ic_action_settings"
        app:showAsAction="ifRoom" />
</menu>
```

Zdrojový kód 13. Ukázka nastavení Action Bar

Zbytek obrazovky je rozdělen do dvou částí: nový průzkum a přechozí průzkum. Horní díl slouží pro zadávání hodnot potřebných pro měření: čas začátku průzkumu, čas konce průzkumu a datum průzkumu. A samozřejmě obsahuje i tlačítko, kterým uživatel začne měření a bude spouštěno nové Activity.

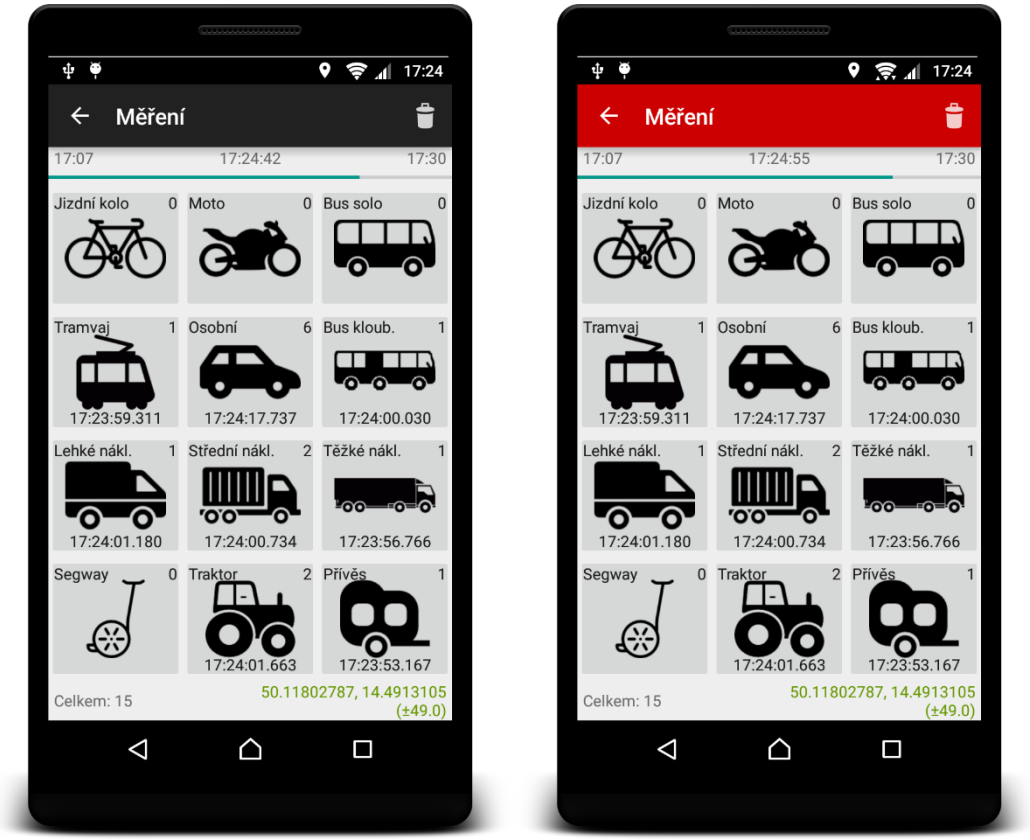
Po klepnutí na datum nebo na čas, uživateli se otevře dialogové okno, ve kterém lze provádět volbu potřebných hodnot. Pro dané účely v operačním systému Android existují speciální prvky, které se nazývají Pickers. Bývají dvojího druhu: Date Picker a Time Picker tj. když přeložíme doslova – sběrač data a sběrač času. Ve své podstatě nabízejí jak vývojářům, tak i uživatelům pohodlný a jednoduchý způsob zadávání hodnot bez nuzností vyvíjet vlastní dialogová okna.

Prvky jsou umístěné pomocí svislého a vodorovného Linear Layout rozložení, která jsou navzájem vnořené. Pro zobrazení textu jsou použity TextView prvky a pro tlačítka, včetně volby data a časů, běžné prvky Button. V souladu s doporučením společnosti Google, které bylo vydáno za účelem sjednocení vzhledu všech aplikaci, volil jsem odpovídající barvy a vzhled prvků.

Dolní část obrazovky je ponechána pro práci s již proběhlými průzkumy. Jednotlivá měření lze vybírat prostřednictvím rozevírací nabídky, realizované s použitím prvku Spinner, ve které jsou zobrazovány ve formátu: datum, čas začátku a čas konce. Zvolený záznam lze buď odstranit anebo exportovat do souboru XML.

5.1.2 Obrazovka pro průzkum

Dané Activity tvoří nejvýznamnější část aplikace, se kterou uživatel se bude nacházet ve styku nejvíce času. Obrazovka (viz Obrázek 11) musí obsahovat, především, tlačítka pro zaznamenávání výskytu události a jejich následné zaznamenávání klepnutím.



Obrázek 11. Ukázka rozložení obrazovky pro průzkum

Zdroj: autor

Vycházel jsem, v první řádě z rozdělení, které bylo popsáno v kapitole 2.2.2 a přesněji řečeno – inspiroval jsem se sčítacím listem celostátního sčítání dopravy, který obsahuje třináct různých kategorii. Ale nepodařilo se mi dosáhnout optimálního umístění třinácti tlačítek na obrazovce obdélníkového tvaru. Proto rozhodl jsem, že bude výhodně celkový počet kategorií zmenšit na dvanáct.

Jenomže když prozkoumáme všechny druhy vozidel na sčítacím listu (viz Obrázek 2. Vzor sčítacího listu), můžeme zjistit určitou souvislost, která pomůže redukovat potřebný počet tlačítek: střední nákladní, těžká nákladní vozidla a traktory mají vlastní kategorií, pokud mají přívěs. A napadlo mě vyčlenit zvlášť tlačítko pro přívěs. Podobná varianta umožní nahradit tři druhy jedním.

Důležité je, že kategorizace zůstane zachována a ušetříme místo na obrazovce. Avšak místo dvanácti tlačítek, jsem obdržel jedenáct. Daný problém byl vyřešen mnohem snadněji – přidáním další kategorie podobě vozítka Segway.

Kromě toho bylo nutné určit vzhled samotných tlačítek. V těsné blízkosti k tlačítku, nebo vevnitř samotného tlačítka, je nezbytné mít název dopravního prostředku. Tím lze zlepšit uživatelskou přívětivost, neboť chtěl jsem, aby uživatel byl schopen používat aplikaci bez nutnosti nastudování dokumentace. Předpokládám, že tím se sníží pravděpodobnost výskytu chyb.

Taktéž pro lepší zpětnou odezvu, uživatel musí mít možnost sledovat stav jednotlivých počitadel, aby byl jist ve funkčnosti aplikace a ve správnosti průběhu měření. Textové pole, které obsahuje číselnou hodnotu počtu stisknutí příslušného tlačítka, je umístěno v pravém horním rohu.

Na spodní straně je umístěno textové pole s poslední hodnotou času, kdy byla naposledy zaznamenána událost stisknutím tlačítka. Tj. nad tlačítkem je umístěná všechna informace potřebná pro měření, a která bude následně uložena.

Horní nabídka Action Bar neobsahuje žádné menu s předvolbami a pro možnost mazání dat bylo zapotřebí umístit nabídku, která přepne rozhraní do režimu odstraňování záznamu. V případě, že uživatel bude potřebovat záznam odstranit, stačí přepnout se do režimu mazání a pak klepnout na příslušné tlačítko kategorie jednou nebo vícekrát – záleží na tom, kolik záznamu bylo zaznamenáno chybně.

Z důvodu blízkého umístění horní řady tlačítek k horní nabídce předpokládal jsem, že může docházet k nežádaným klepnutím na tlačítka z Action Baru a obdobný problém je na dolní části obrazovky u navigační lišty. Proto za účelem zvětšení vzdálenosti mezi blokem tlačítek a příslušnými lištami, bylo rozhodnuto umístit do uvolněného prostoru prvky s informací. Nahoře přibyl řádek s časovými údaji o průzkumy: čas začátku, aktuální čas a čas konce. Taktéž časový průběh je předveden prostřednictvím prvku Progress Bar. Dolní část stala obohacena o informaci o celkovém počtu zaznamenaných událostí (počet projetých vozidel) a informace o aktuální poloze.

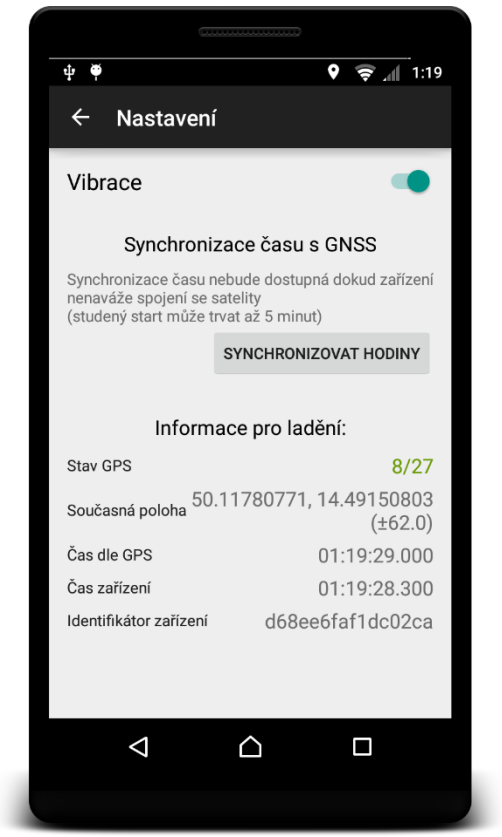
5.1.3 Obrazovka s předvolbami

Poslední Activity representuje rozhrání pro nastavení předvoleb aplikace. První položka odpovídá za přepínání funkce vibrování při doteku tlačítek na obrazovce pro průzkum.

Dále, jak bylo zmíněno v kapitole 2, jeden z požadavků je možnost synchronizace hodin. Danou funkci lze implementovat pomocí GNSS, který poskytuje velmi přesné údaje o aktuálním čase a v dostatečné míře postačí pro účely dopravních průzkumů.

Takže následující část obrazovky je věnovaná účelům synchronizace hodin. Tlačítko nebude aktivní, dokud zařízení nenaváže spojení s družicemi GNSS a nezíská aktuální polohu.

Zbytek obrazovky obsahuje informaci pro ladění, které napomohou v případě, že zařízení delší dobu nezíská aktuální polohu nebo pro porovnání hodin zařízení s hodinami GNSS a rovněž i pro identifikaci zařízení, ze kterého byl uložen soubor s hodnotami měření.



Obrázek 12. Ukázka rozložení obrazovky s předvolbami

Zdroj: autor

5.2 Ukládání dat

Pro ukládání dat v rámci jedné aplikace, nebo celého systému, je nám k dispozici následující způsoby:

- Sdílená nastavení – umožňuje ukládat data ve formátu klič-hodnota v rámci jedné aplikace.
- Vnitřní úložiště – taktéž poskytuje prostor pro ukládání soukromých dat v rámci jedné aplikace pomoci souborů, jejichž obsah a způsob uložení vývojář může určit sám.
- Externí uložiště – na rozdíl od předchozí varianty, k datům, uloženým na externím uložišti, mají, přistup ostatní aplikace, nainstalované v systému, a taktéž i uživatel.
- Databáze SQLite poskytuje prostor pro ukládání strukturovaných dat do soukromé databáze.
- Vzdálené ukládání – jedná se o klient-serverový způsob ukládání dat na vlastní server.

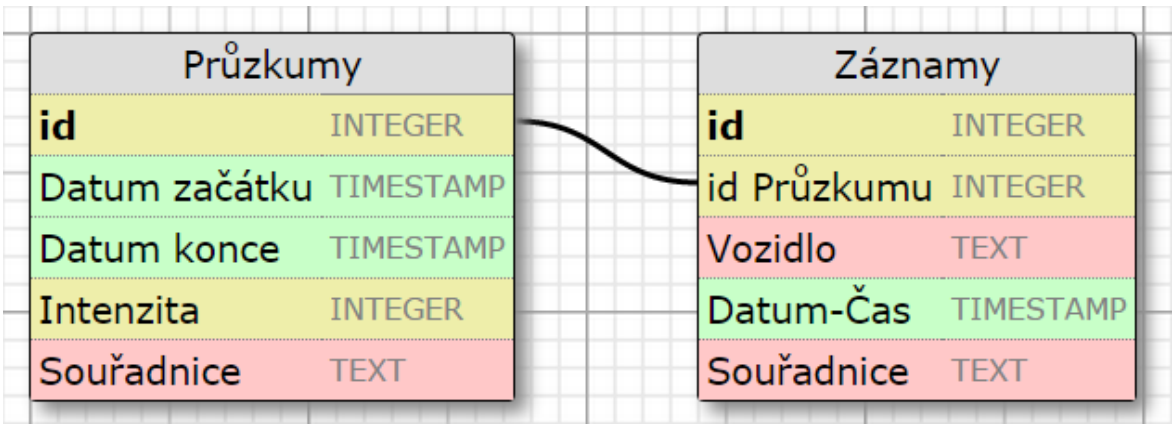
V aplikaci byly využity tři z pěti výše uvedených způsobů. Ukládání do vnitřních pamětí omezuje přístup k souborům z externích aplikací a neumožňuje pomocí vestavených komponent operačního systému soubory sdílet. Každopádně je zapotřebí ukládat soubor s výsledky průzkumu na externí uložiště a nevidím důvod, aby data byla uložená na vnitřní paměti. Taktéž v dané fázi vývoje není využita možnost ukládání na vzdálené servery, ale pokud o aplikace najde uplatnění v praxi, realizace komunikace se serverem je jenom otázka času.

Do sdílených nastavení jsou uložená jenom data o stavu přepínače nebo dostupnosti vibrace při dotyku tlačítek.

Externí úložiště se používá pro ukládání souboru XML, který obsahuje záznamy s jednotlivého průzkumu a následně tento soubor může byt sdílen jinými nainstalovanými aplikacemi, např. poslán v příloze elektronickou poštou, nahrán do oblačního úložiště, sdílen prostřednictvím Bluetooth nebo NFC aj.

5.2.1 Databáze

Záznamy tvoří větší část ze všech dat, která aplikace ukládá, jsou strukturovaná, a proto bylo rozhodnuto o jejich ukládání do databáze. Prostřednictvím SQLite je možné snadno data dohledávat, aplikovat různé filtry a samozřejmě je mazat. Při prvním návrhu jsem uvažoval jenom o jedné tabulce, ale jelikož je zapotřebí rozlišovat příslušnost záznamu jednotlivým průzkumům, v jedné tabulce by docházelo k redundanci identifikátoru průzkumu. Proto jsem vyčlenil zvlášť tabulku, která bude obsahovat data o průzkumu a zvlášť tabulku se záznamy (viz Obrázek 13). Do první se zaznamenává identifikátor průzkumu, který je klíčem, hodnoty začátku a konce průzkumu, počet záznamu a průměr ze souřadnic příslušného průzkumu tj. průměrná poloha.



Obrázek 13. Struktura použité databáze

Zdroj: autor

Druhá tabulka je naplněná daty s identifikátorem záznamu (primární klíč), identifikátor průzkumu je cizím klíčem, který určuje příslušnost záznamu určitému průzkumu z předchozí tabulky, druhem vozidla, časovým razítkem a polohou, kde byla událost zaznamenána.

5.3 Realizace části programování

S programováním v jazyce Java, ani s programováním pro operační systém Android, jsem se nikdy neměl příležitost setkat. Vyžádalo to velkou časovou náročnost a poměrnou část práci jsem musel věnovat samostudiu a doplňování chybějících znalostí.

V předchozí podkapitole jsem již zmiňoval, že každá obrazovka aplikace se nazývá Activity. Activity má svůj životní cyklus, přičemž při přechodech z jednotlivých stavů můžeme pomoci metod provádět určité operace. Tak, například, při spuštění Activity bude vykonán zdrojový kód, obsazený v metodě onCreate() a při pozastavení – v metodě onPause(). Podobným způsobem je umožněno provádět vlastní úkony za různých okolností.

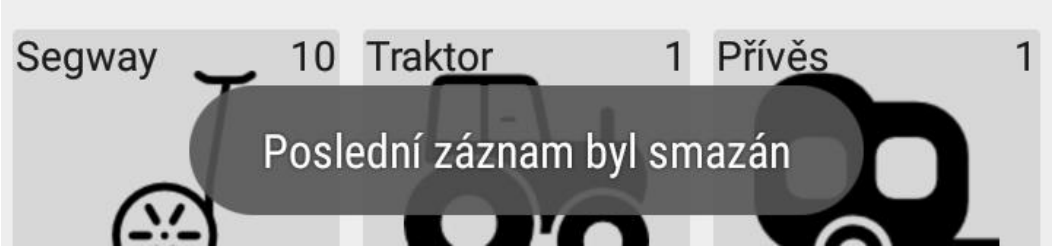
Po skončení jedné Activity často je zapotřebí poslat data do následující. V mém případě při přechodu z úvodní obrazovky na obrazovku určenou pro měření je nutné poskytnout data o začátku a konce měření. Pro sdílení malého množství hodnot je nám k dispozici metoda putExtra() třídy Intent. Objekt Intent používáme při požadování reakce od jiných komponent aplikace. Umožňuje nejenom spouštět Activity a služby, ale taktéž i posílat do nich data.

V zdrojovém kódu se v přílohách nebo uvedeném v textu bude vyskytovat objekt Log. Jedná se o API pro ukládání informaci obsazené v parametrech do žurnálu činnosti. Například následující ukázka zdrojového kód uloží řetězec symbolů do žurnálu ladění (debug) s tágem, obsazeným v proměnné TAG.

```
Log.d(TAG, "Událost proběhla úspěšně");
```

Zdrojový kód 14. Příklad použití objektu Log

Dalším způsobem pro zobrazování zpráv je použití objektu Toast. Jedná se o nejjednodušší možnost zobrazení informaci , která je založená na dialozích v OS Android. Zobrazí zprávu v dolní části obrazovky a následně po několika sekundách zmizí. Na obrázku Obrázek 14je znázorněna krátká Toast zpráva s příslušným zdrojovým kódem.




```
Toast.makeText(this, "Poslední záznam byl smazán",
                Toast.LENGTH_SHORT).show();
```

Obrázek 14. Přiklad krátké Toast zprávy s příslušným zdrojovým kódem

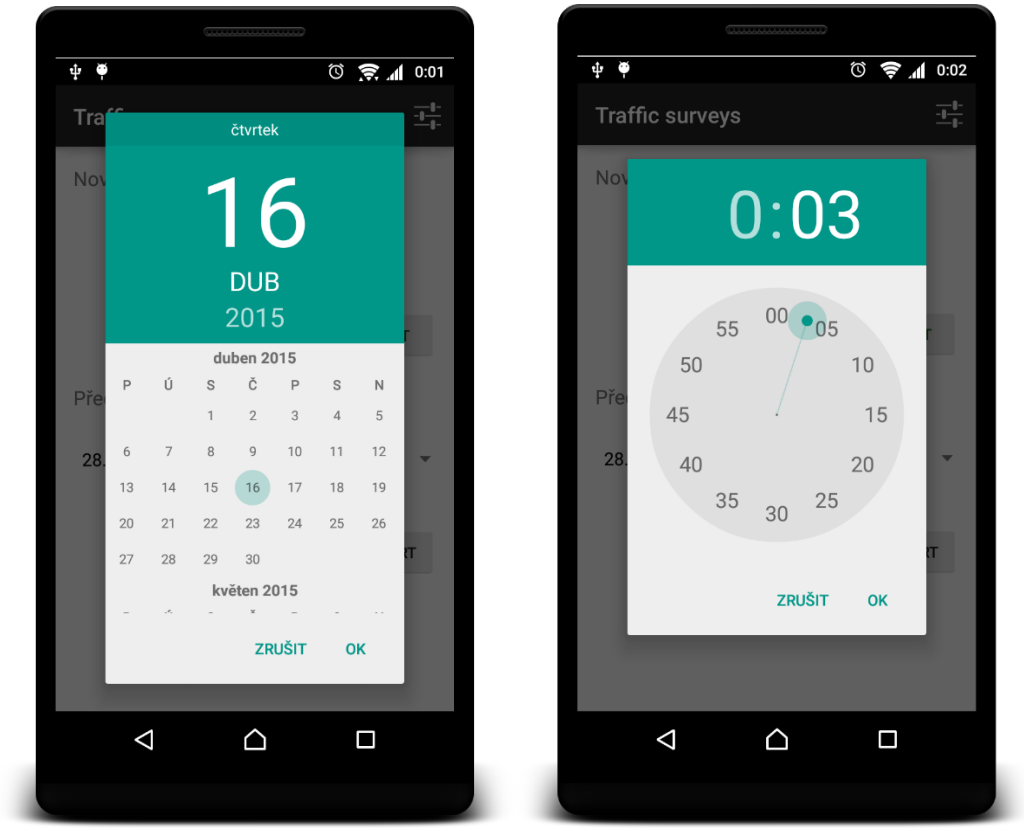
Zdroj: autor

5.3.1 Úvodní obrazovka

Na úvodní obrazovce (viz obrázek Obrázek 10. Rozložení úvodní obrazovky), se nacházejí prvky pro zvolení data a času, volbu operaci s předešlými průzkumy a tlačítko pro přechod do nastavení. Nadpisy jsou zastoupeny statickými hodnotami, které jsou uloženy v souboru s rozložením. Provádět změny jejich obsahu nebo vzhledu, zatím není potřeba a proto nejsou inicializovány v souboru MainActivity.java, ve kterém se nachází zdrojový kód dané Activity.

5.3.1.1 Nový průzkum

Při spuštění uživatel potřebuje nastavit hodnoty začátku a konce průzkumu. Mobilní operační systémy z hlediska propracovanosti uživatelského rozhraní jsou mnohem pokročilejší v porovnání se svými staršími bratří, běžícími na stolních počítačích. Umožňují se zbavit složitého zadávání dat s určováním formátování v závislosti na různých předvolbách systému a jazyka. Poskytuje danou funkcionalitu do jisté míry samostatná entita, která ale žije spolu s Activity s názvem dialog. Přesněji pro volbu data slouží DatePickerDialog a pro volbu času – TimePickerDialog, pomocí kterých je zprostředkována příjemnější interakce uživatele s aplikací. Na snímcích obrazovky na obrázkuObrázek 15jsou zobrazeny příslušné dialogy.



Obrázek 15. Vzhled dialogů DataPickerGialog (vlevo) a TimePickerDialog (vpravo)

Zdroj: autor

Dialogy jsou vyvolány procedurou onCreateDialog, do které předáme identifikátor dialogu. Je to sice způsob zastaralý a v posledních verzích Androidu je doporučeno používat tzv. fragmenty [28], ale během vývoje fragmenty způsobovali komplikace. Především vyvolávaly pád aplikaci za následujících podmínek:

- na obrazovce musí byt zobrazen aktivní dialog DatePickerDialog nebo TimePickerDialog,
- musí dojit k otočení zařízením příp. k vyvolání události pro překreslení aktuální Activity.

Nepodařilo se mi vypátrat zdroj podobného chování, a proto jsem byl nucen vyloučit použití fragmentů pro účely zobrazení dialogy s volbou data nebo času.

Při spuštění Activity proměnná začátku průzkumu je nastavena na aktuální čas a datum. Konec průzkumu je ve výchozím nastavení posunout o jednu hodinu dopředu. Výchozí délka průzkumu taktéž činí jednu hodinu a po volbě času začátku pomoci dialogu TimePickerDialog čas konce je vždy posunout o výchozí délku. O implementaci volby data konce měření jsem uvažoval, ale nepředpokládám, že během použití aplikace vyskytne potřeba sledovat změny dopravy přes 24 hodin pomoci ručního měření. Pokud by se vyskytl podobný požadavek,

hlavními omezujícími faktory budou jak schopnost osoby provádějící průzkum zaznamenávat data v průběhu dvou desítek hodin, tak i především baterie zařízení, která v běžném chytrém mobilu nebo tabletu nezaručí dostatečnou výdrž pro tento účel. Proto uživateli je umožněno zadávat jenom čas konce a aplikace následně porovná čas začátku s časem konce průzkumu, a pokud konec měření je před začátkem, přičte k datu měření jeden den.

Tlačítko pro začátek průzkumu je umístěno pod oddílem s volbou času a při jeho stisknutí proběhne nejenom spuštění dalšího Activity. Ale zaprvé musíme uložit data do databáze. Z dialogu pro zadávání časů a data získáváme hodnoty začátku a konce průzkumu a ukládáme je do příslušné tabulky. Výsledkem operace bude hodnota identifikátoru průzkumu (viz Zdrojový kód 15. Přiklad uložení dat o průzkumu do databáze).

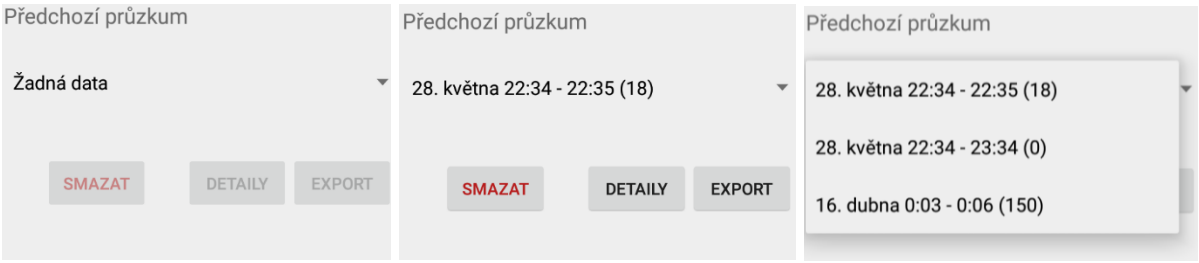
```
cvPruzkum.put(ZACATEK, calBegin.getTimeInMillis());
cvPruzkum.put(KONEC, calEnd.getTimeInMillis());
cvPruzkum.put(INTENZITA, 0);
cvPruzkum.put(SOURADNICE, "N/A");
// insert data
Log.d(TAG, "data prepared");
long rowID = dbPruzkumy.insert("pruzkumy", null, cvPruzkum);
Log.d(TAG, "data inserted with id = " + rowID);
```

Zdrojový kód 15. Přiklad uložení dat o průzkumu do databáze

Získaný identifikátor uložíme a spolu s hodnotami začátku a konce průzkumu pošleme do další Activity (viz přílohu Zdrojový kód A. 1), které je věnována podkapitola 5.3.2.

5.3.1.2 Předchozí průzkum

Dolní část obrazovky je věnovaná operacím s již naměřenými daty. Volbu předchozího průzkumu zajišťuje prvek Spinner, který po kliknutí zobrazí rozevírací nabídku obsahující data z databáze (viz Obrázek 16). Při spuštění Activity z tabulky průzkumu budou získána data a jednotlivých měření. Následně jsou převedeny do řetězce, který obsahuje údaje o začátku průzkumu, konce průzkumu, a počtu záznamů ve formátu v souladu s místními předvolbami zařízení.



Obrázek 16. Ukázka různých stavu prvku Spinner

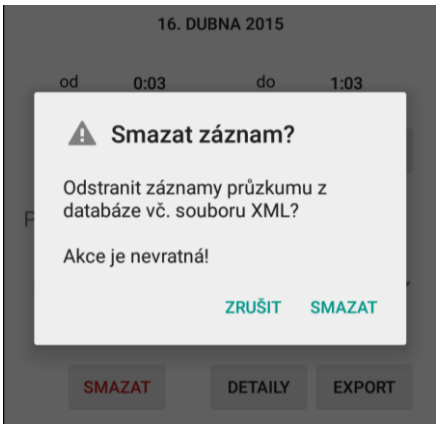
Zdroj: autor

Pokud by tabulka průzkumů neobsahovala žádná data, a to z důvodu, že se pomocí aktuálního zařízení žádné měření neprovádělo, anebo v případě že uživatel odstranil všechny záznamy, prvek Spinner zobrazí jenom informaci o chybějících datech a samotný prvek, včetně se níže nacházejících tlačítek, se přepne do neaktivního modu.

Když aplikace nalezne v databázi záznamy o předchozích průzkumech, uživateli se nabídne možnost provést se zvoleným průzkumem jednu z třech akcí prostřednictvím odpovídacích tlačítek:

- smazat,
- detaily,
- export.

V případě, že si uživatel rozhodne odstranit vše záznamy z příslušného průzkumu, stačí jenom klepnout na první tlačítko. Vytvoří se novy objekt AlertDialog, který je také z rodiny dialogů a podoben DatePickerDialog nebo TimePickerDialog, sloužící se pro zobrazení většího objemu informaci a obsahuje totiž možnost volby následující operaci. Aplikace se zeptá uživatele, jestli on stále má úmysl data odstranit, a nabídne možnost uzavření dialogu s odstraněním záznamů anebo se zrušením operace (viz příloha Zdrojový kód A. 2). Snímek obrazovky zachycuje následující obrázek:



Obrázek 17. Prvek AlertDialog s varováním o odstranění záznamů

Zdroj: autor

Jestli uživatel rozhodné ve svém záměru pokračovat, budou data z databáze odstraněna a to následujícím způsobem:

1. uloží se poloha zvoleného průzkumu z rozevírací nabídky prvku Spinner, která odpovídá číslu řádku v tabulce,
2. provede se pokus o odstranění souboru XML z externího uložiště,

3. dle polohy se z tabulky průzkumů získá příslušný identifikátor (klíč),
4. odstraní se záznam z tabulky průzkumů s odpovídajícím klíčem,
5. z tabulky záznamů budou odstraněny všechny řádky s odpovídajícím cizím klíčem ve sloupci s identifikátory průzkumů.
6. Zobrazí se Toast dialog se zprávou o úspěšném výsledku operace.

Všechny změny jsou nevratné o čem uživatel informován pomoci předchozího dialogu.

Že by uživatel rozhodl záznamy neodstraňovat či databáze stále není prázdná, nabízí se prostřednictvím prvku Button s názvem „Detaily“ možnost zobrazení informaci o zvoleném průzkumu. Podrobnosti jsou zobrazeny taktéž v dialogu, který je vytvořen po klepnutí na příslušné tlačítko. Do zprávy objektu AlertDialog je umístěna informace z tabulky průzkumů (začátek, konec, průměrná poloha a počet událostí) a seznam všech druhu událostí s odpovídající četností z tabulky záznamů. Zjišťování daných hodnot z databáze je v porovnání s mazáním složitější:

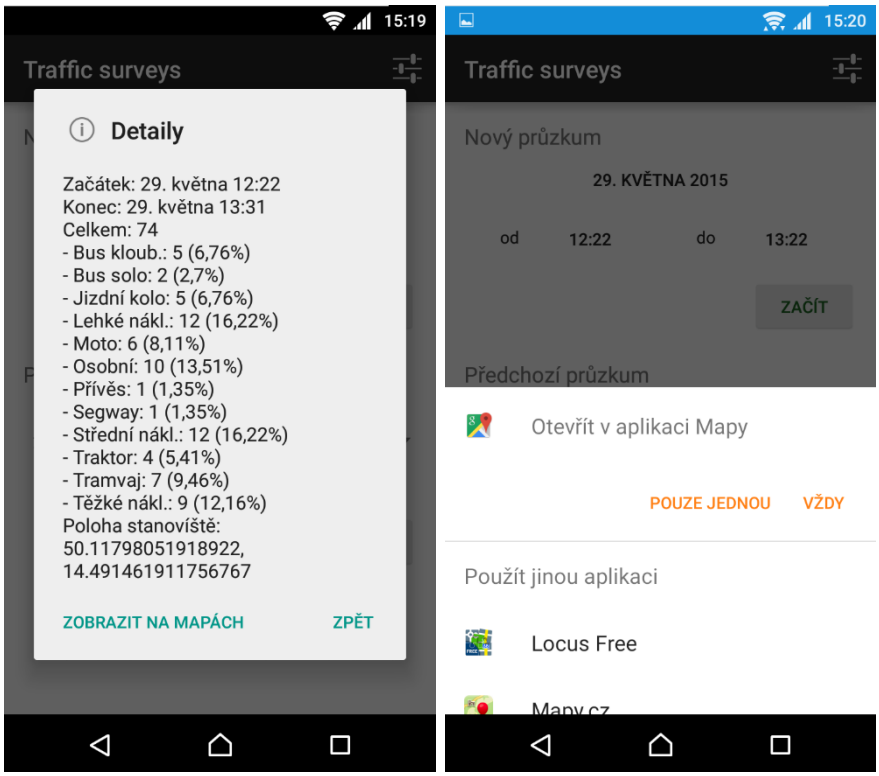
1. uloží se poloha zvoleného průzkumu z rozevírací nabídky prvku Spinner, která odpovídá číslu řádku v tabulce,
2. dle polohy se z tabulky průzkumů získá příslušný identifikátor (klíč), hodnoty začátku a konce průzkumu a poloha,
3. z tabulky záznamů se získají všechny druhy události,
4. postupně pro každý druh události se zjistí její četnost.

Body č. 3 a č. 4 z výše uvedeného algoritmu v aplikaci implementován následujícím způsobem:

```
Cursor c2 = dbPruzkumy.query(TABLE_ZAZNAMY,
        new String[]{VOZIDLO},
        ID_PRUZKUMU + " = " + pos,
        null,
        VOZIDLO, null, null);
List<String> SCategories = new ArrayList<String>();
List<Integer> Pocty = new ArrayList<Integer>();
if (c2.moveToFirst()) {
    do {
        String sCategorie = c2.getString(c2.getColumnIndex(VOZIDLO));
        SCategories.add(c2.getString(c2.getColumnIndex(VOZIDLO)));
        Pocty.add((int) DatabaseUtils.longForQuery(dbPruzkumy,
                "SELECT COUNT(*) FROM " + TABLE_ZAZNAMY +
                " WHERE " + ID_PRUZKUMU + " = " + pos +
                " AND " + VOZIDLO + " = '" + SCategories.get(i) + "'"
                , null));
        i++;
    } while (c2.moveToNext());
}
```

Zdrojový kód 16. Ukázka algoritmu získávání z databáze dat o událostech a jejích četnostech

Následně, pokud se podařilo získat údaje o přibližné poloze stanoviště, bude připraven textový řetězec URI pro zobrazení polohy ve formátu: $geo{:}LAT,LONG?q=LAT,LONG$, kde LAT – zeměpisná šířka a $LONG$ – zeměpisná délka. Taktéž AlertDialog je doplněn o tlačítko, které spustí nové Activity. V případě, že v operačním systému vyskytuje jedna nebo více aplikaci schopných zpracovat dané URI, bude spuštěn příslušný software anebo uživatel bude moct si zvolit aplikaci, pomocí které URI bude zpracováno. Na Obrázek 18 jsou snímky obrazovky s informací o průzkumu a volbou aplikace pro zobrazení polohy.



Obrázek 18. Ukázka dialogu s informací o průzkumu a nabídka volby aplikace pro zpracování URI se souřadnicí

Zdroj: autor

Pokud během průzkumu od služby zjišťování polohy nebude možné získat souřadnice, do sloupce se souřadnicemi bude uložena hodnota N/A a v dialogu nebude zobrazovaná možnost zobrazování na mapách.

Ukládání a následné sdílení lze provést dotykem tlačítka „Export“. Ve složce „průzkumy“ na externí paměti bude vytvořen XML soubor. Do kořenového elementu s názvem „Průzkum“ jsou vnořeny elementy „Vlastnosti“ a „Záznamy“. První obsahuje elementy s průměrnou polohou, začátkem měření, koncem měření a unikátním identifikátorem zařízení, ve kterém soubor byl vytvořen. V záznamech se nachází element „událost“, obsahující elementy s druhem události,

časovým razítkem a souřadnicemi. Struktura je znázorněna ve Zdrojový kód 17 a ukázka vzorového souboru je v příloze Zdrojový kód A. 1.

```
<Průzkum>
    <Vlastnosti>
        <Lokalita>
            <souřadnice></souřadnice>
        </Lokalita>
        <Začátek>
            <datum></datum>
            <čas></čas>
            <pásmo></pásmo>
        </Začátek>
        <Konec>
            <datum></datum>
            <čas></čas>
            <pásmo></pásmo>
        </Konec>
        <devID></devID>
    </Vlastnosti>
    <Záznamy>
        <událost>
            <Vozidlo></Vozidlo>
            <Datum></Datum>
            <Čas></Čas>
        </událost>
    </Záznamy>
</Průzkum>
```

Zdrojový kód 17. Struktura XML, dle které aplikace generuje výstupní soubor

Po skončení vlákna vytváření souboru aplikace nabídne uživateli tento soubor sdílet prostřednictvím standartní Activity, která jíž byla použita pro zobrazení souřadnic na mapách. Kromě samotného URI s cestou k souboru, taktéž obsahuje informaci o jeho typu. Mezi podporované aplikace, například, patří poštovní klienty, Google Drive, sdílení pomocí Bluetooth nebo NFC.

V liště Action Bar se nachází tlačítko s ikonou, dotyk které přesměruje uživatele na obrazovku s předvolbami.

5.3.2 Obrazovka pro průzkum

Activity určená pro zaznamenávání události může být spouštěna pouze tlačítkem z úvodní obrazovky. V první řadě je nastaven parametr pro zabránění zhasnutí obrazovky, aby uživatel nemusel měnit systémová nastavení. Při startu obdržíme na vstupu údaje o začátku a konce průzkumu, které slouží pro vytváření prvku ProgressBar, jehož maximální hodnota činí:

$$PB_E = T_E - T_B \tag{11}$$

kde: PB_{max} maximální hodnota prvku ProgressBar,

T_E čas konce průzkumu

T_B čas začátku průzkumu

Taktéž při spuštění Activity bude vytvořeno vlákno, které slouží čtyřem účelům:

- aktualizace textu v prvku TextView s aktuálním časem,
- nastavení aktuální hodnoty prvku ProgressBar,
- těsně před začátkem a po skončení průzkumu přehraje příslušné zvuky.

Aktuální hodnotu pro prvek ProgressBar spočítáme podle podobného vztahu, jako je rovnice (11), ale místo času konce průzkumu dosadíme aktuální čas:

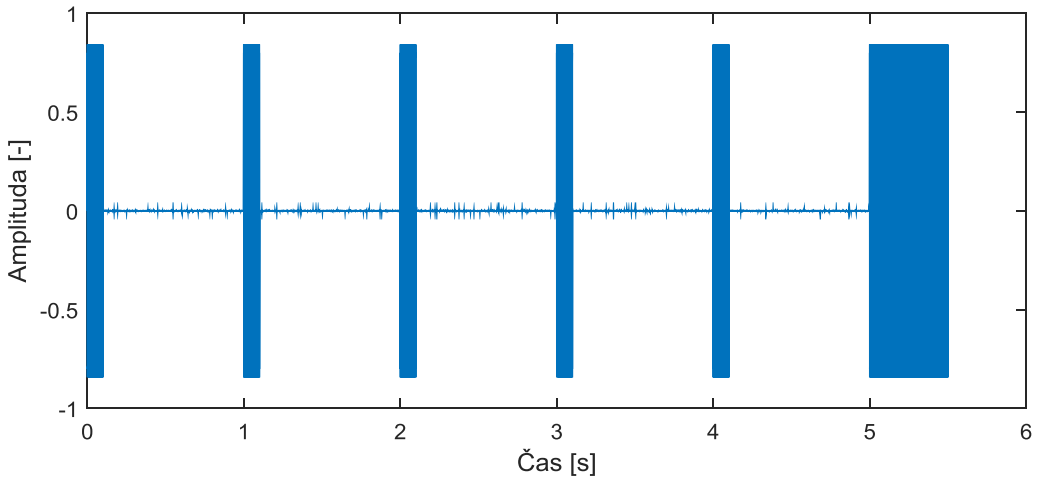
$$PB_N = T_N - T_B \tag{12}$$

kde: PB_N aktuální hodnota prvku ProgressBar,

T_N aktuální čas

T_B čas začátku průzkumu

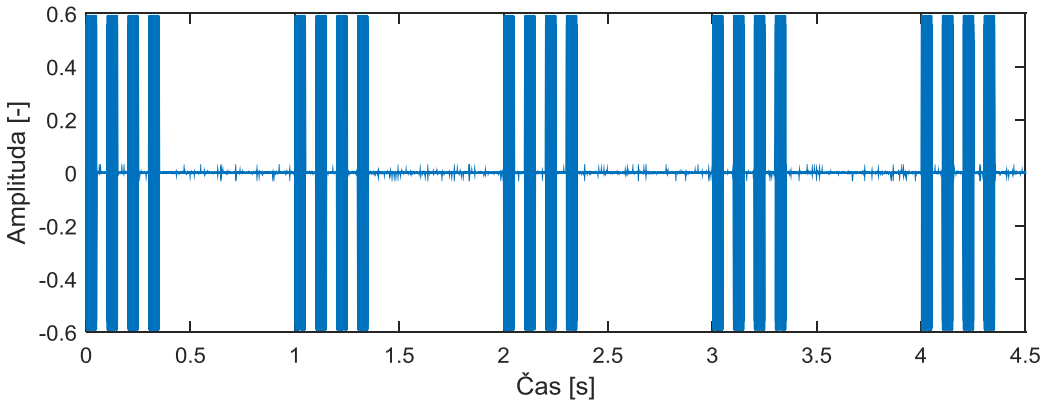
Rozhodl jsem, že když začátek a konec průzkumu bude provázet i zvuk – usnadní to měření a uživatel s větší pravděpodobností začne měření včasně. Pro zvuk začátku jsem se inspiroval zvukem greenwichského času (Greenwich Time Signal) [33], který se celosvětově používá v rozhlasu pro kalibraci hodin. Je složen z pěti krátkých pípnutí s frekvenci $1\ kHz$ o délce trvání $0{,}1\ s$ a jednoho delšího trvající $0{,}5\ s$, které jsou rozděleny pauzami s trváním $0.9\ s$. Graf je znázorněn na obrázkuObrázek 19. Se zvukem taktéž je synchronizovaná vibrace. Zvuk se začíná přehrávat za pět vteřin před začátkem průzkumu.



Obrázek 19. Graf znázorňující zvuk začátku průzkumu

Zdroj: autor

Zvuk pro ukončení byl použit v podobě sérii třech pípnutí o frekvenci $2\ kHz$. Pauzy v sérii jsou o délce $0{,}05\ s$. Mezi sériemi pauza činí $0{,}65\ s$. Na obrázku č. XX je k nahlédnutí graf zvuku. Analogicky zvuku začátku při přehrávání zvuku na konce spouští se vibrace.



Obrázek 20. Znázornění zvuku konce průzkumu

Zdroj: autor

Po spuštění se načítají hodnoty se sdíleného nastavení ShredPreferences. Je zapotřebí zjistit, jestli v předvolbách je povolena funkce vibrace.

Dále inicializuje se objekt poskytující informaci o poloze (viz přílohu Zdrojový kód A. 4). Nastavují se zdroj dát: jestli pro určování polohy budou použity přístupové body bezdrátové sítě a poloha vysílacích stanic mobilní sítě nebo budou použity služby GNSS, které jsem i zvolil. Pokud služba určování polohy nemůže poskytnout aktuální pozici, v dolním prvku TextView se zobrazí nadpis „N/A“ červené barvy. Po obdržení správných dat s polohou, zobrazí se aktuální souřadnice spolu s přesností zelenou barvou. U prvku může dojít ke změně barvy na oranžovou v případě dočasné nedostupnosti signálu z družic

Na horní liště se nachází tlačítko pro přepínání řežím odstranění záznamů. Po jeho zapnutí se změní barva horní lišty na červenou, aby uživatel měl možnost jednoduše poznat, jestli data budou zaznamenávána nebo odstraňována. A následně je nutné provést volbu události, kterou je potřeba odstranit. Taktéž tlačítko nebude zobrazeno, dokud aplikace nezaznamená alespoň jeden výskyt události. Na následujícím obrázku jsou znázorněny všechny stavy lišty ActionBar.

Obrázek 21. Ukázka různých stavu horní lišty ActionBar

Zdroj: autor

Pokud uživatel se dotkne jakéhokoliv tlačítka s ikonami kategorií událostí, zaprvé se zaznamená čas a pro záznam události bude vyvolána metoda updateData, do které předáme časové razítko, objekty tlačítka a nadpisů na něm a řádkem se sloupcem, kterému přísluší. Následně lze rozdělit průběh na dva případy: režimu zaznamenávání záznamu a řežím odstraňování záznamu. Společnou mají jenom část, týkající se aktualizaci nadpisů na příslušném tlačítku tj. prvků TextView, které ve skutečnosti překrývají jeho a hodnota celkového počtu záznamů.

Při zaznamenávání dat, tabulka záznamů bude doplněná o nový řádek obsahující aktuální časové razítko, souřadnice, jestli jsou dostupné, a druh události získaný z prvku TextView, který se nachází na tlačítku. Podobný přístup umožňuje jenom změnou názvu i ikony v příslušném souboru layout dosáhnout určité přizpůsobivosti aplikace pro různé průzkumy a kategorie událostí.

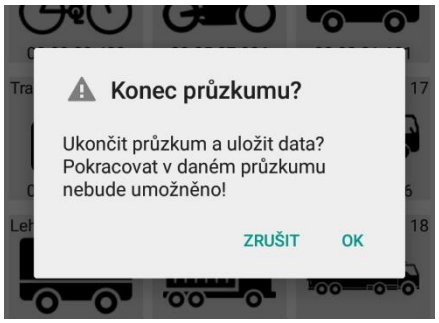
Při odstranění záznamu se ze stejnojmenné tabulky získají řádky odfiltrované podle hodnoty ve sloupci druhu události s kategorií odpovídající názvu tlačítka a seřazené sestupně podle klíče. Z prvního řádku bude získaná hodnota identifikátoru. Z druhého řádku bude získán časový údaj, který po odstranění poslední události musí být zobrazen na tlačítku a proběhne odstranění prvního řádku. V případě úspěšnosti operace bude zobrazena Toast zprava, v opačném případě obdobně zobrazíme uživateli zprávu o tom, že žádné záznamy odpovídající kategorií tlačítka neexistují. Implementace algoritmu odstranění je následující:

```
Cursor cPruz = dbPruzkumy.query(TABLE_ZAZNAMY,
        new String[]{ID, ID_PRUZKUMU, VOZIDLO, TIMESTAMP},
        ID_PRUZKUMU + " = " + id + " AND " + VOZIDLO + " = '" + sCat+"'",
        null, null, null, ID + " DESC", "1");
cPruz.moveToFirst();
long idDelete = cPruz.getLong(cPruz.getColumnIndex(ID));
Log.d(TAG, "Get id to delete " + idDelete);
timestamp = cPruz.getLong(cPruz.getColumnIndex(TIMESTAMP));
Log.d(TAG, "Get timestamp to delete " + timestamp);
if (cPruz.moveToNext()) {
    timestamp = cPruz.getLong(cPruz.getColumnIndex(TIMESTAMP));
} else {
    timestamp = 0;
}
dbPruzkumy.delete(TABLE_ZAZNAMY, ID + " = " + idDelete, null);
```

Zdrojový kód 18. Ukázka algoritmu odstranění chybného záznamu

Teď v obou dvou režimech bude vykonána aktualizace prvků TextView příslušného tlačítka a hodnoty celkového počtu výskytu událostí, která nahradí v tabulce průzkumů staré číslo. Taktéž spočítáme průměrnou polohu na základě všech hodnot v tabulce záznamů pro současný průzkum a její hodnotou aktualizujeme tabulku průzkumů.

Dále jsem se pokusil minimalizovat šanci nechtěného opuštění obrazovky s průzkumem ukončením příslušné Activity. Při pokusu o ukončení Activity pomocí softwarového nebo hardwarového tlačítka „zpět“ bude zobrazen dialog AlertDialog, který je znázorněn na Obrázek 22, prostřednictvím kterého uživatel bude varován o tom, že pokud on opustí obrazovku, pokračovat v současném průzkumu nebude umožněno. Ale před tím, než Activity bude ukončená, provede se aktualizace času ukončení průzkumu v tabulce průzkumů.



Obrázek 22. Prvek AlertDialog zabraňující náhodnému opuštění obrazovky pro měření

Zdroj: autor

5.3.3 Předvolby

Poslední Activity, která je reprezentovaná v aplikací obrazovkou s předvolbámí je poměrně jednoduchá a není tak složitá, jako ostatní. Při její startu se načtou hodnoty ze sdíleného nastavení a v závislosti na ních bude nastaven přepinač funkce vibrace, ale pouze v případě, že zařízení to umožňuje, jinak prvek Switch nabyde atributu neaktivní. Po změně stavu prvku jeho hodnota se uloži zpět do sdileného nastavení.

Dále je inicializovan objekt LocationManager, který ale narozdil od předchozého Activity je obphacen o instanci GpsStatus.Listener, poskytujicí informaci o počtu satelitu, které zařízení vidí a kolik z nich použíto pro zjišťování polohy. Danou informaci budeme zobrazovat v přislušném TextView a menít jeho barvu.

Se synchornizáci hodin nástal ponekud nečenkaný problém. V nastaveních operačního systému lze nalezt funkci pro automaticke nastavení času pomocí dat ze sítě. Android nedovoluje z bezpečnostních důvodů nesystemovým aplikacím nastavovat hodiny. Tohle omezení se dá obejít pouze tzv. rootovaním, které, bohužel, není v souladu se záručnými podminkami. Takže funkci synchronizaci hodin se mi nepodařilo realizovat, a proto tlačitku byla ponechána jenom funkce pro porovnání času ziskaného pomocí GNSS a hodin v systému. Nebo tuhle funkci lze využit pro porovnání hodin mezi různymi zařízenimi. Tlačítko bude neaktivní do té doby, pokud nebudé navazano spojení s družicemi.

Ale narazil jsem na jednou velmi zajimavou závislost – několikanásobné kliknutí na tlačítko umožňuje zmenšít odchylku mezi hodinami na GNSS a v systému. V dokumintaci není vysvětlení daného děje. Předpokladám, že při žadosti o aktualizaci dat stisknutím tlačitka, LocationManager poskytuje táto data celosistemově a oni mohou byt použitá operačním systémém pro zpřesnění hodin.

6 Ověření funkčnosti realizovaného návrhu

Ověření jsem prováděl ze třech hledisek. V první řadě bylo posouzeno splnění požadavků, jestli jsou realizovány všechny funkce, které nejsou implementovány. Dále bylo provedeno vyzkoušení aplikace v praxi a výsledky porovnány s papírovými záznamy. Třetí hledisko je analýza využití baterie a její životnosti při použití aplikace.

6.1 Spotřeba energie aplikací

Analýza životnosti zařízení byla provedena dvěma způsoby: bez spouštěné aplikace a se spouštěnou aplikace. Stanovil jsem následující podmínky pro provedení měření:

- zapnutý režim „v letadle“ s omezením bezdrátové komunikaci,
- délka měření činí dvě hodiny,
- jas obrazovky je maximální,
- zařízení na začátku měření musí být plně nabitá,
- v přístroji musí být zapnuta funkce zjišťování polohy.

K dispozici jsem měl tři zařízení. Dva z nich jsou chytré mobily: Sony Xpreria Z3 Compact, HTV One V a jeden tablet Asus Nexus 7 2013. v následující tabulce jsou uvedeny základní charakteristiky zařízení:

Tabulka 7. Charakteristiky zařízení použitých pro testování aplikace

	Sony Xperia Z3 Compact	HTC One V	Asus Nexus 7 2013
Velikost displeje [palce]	4.6	3.7	7
Rozlišení obrazovky [pixely]	720 x 1280	480 x 800	1920 x 1200
Operační systém	Android 5.0.2	Android 4.2.2	Android 5.1.1
Kapacita baterie [mAh]	2600	1500	3950
Frekvence procesoru [GHz]	2.5	1	1.5
Počet jader	4	1	4
Processor	Qualcomm Snapdragon 801 8974-AC	Qualcomm Snapdragon S2 MSM8255	Qualcomm Snapdragon S4 Pro APQ8064

Ze zařízení byly průběžně získávány hodnoty napětí a úrovně baterie. Ale jelikož nesmějí být připojeny k počítači, protože jinak by docházelo k nabíjení, zařízení musí zaznamenávat jednotlivé stavy samostatně, což také vyžaduje výpočty a zrychluje vybíjení. Na HTC One V je nainstalována alternativní verze firmware bez jakýchkoliv doplňkových aplikací, jelikož výrobce žádné aktualizace obsahující operační systém Android novější, než verzi 4.0 nevydával.

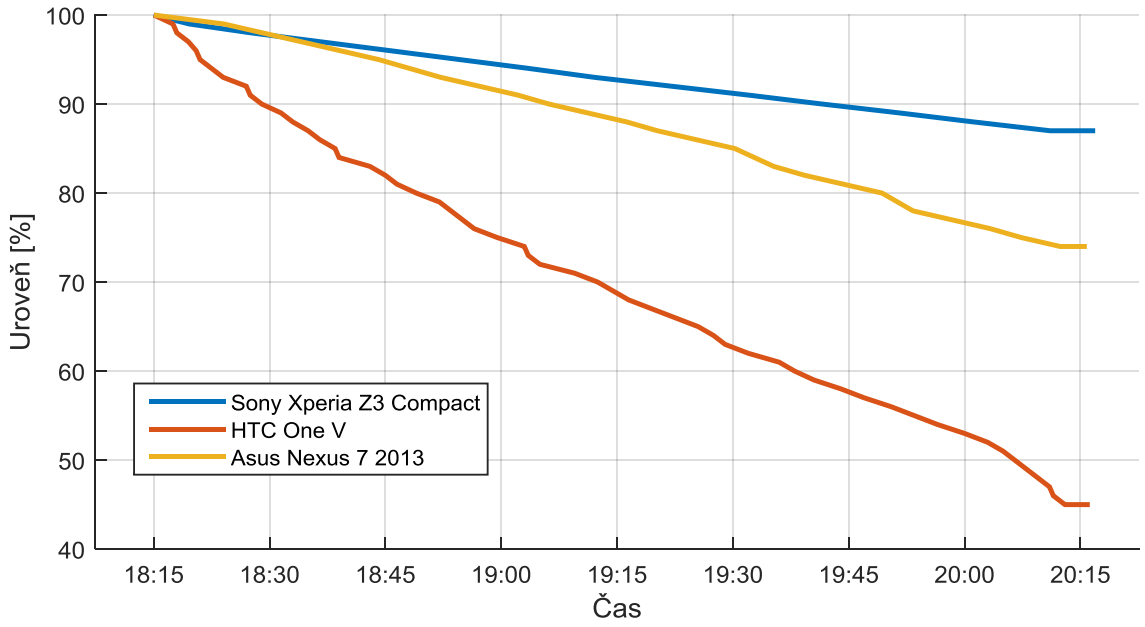
Ostatní dva přístroje běžně se používají a mají nainstalovány různý software od třetích stran, takže na čistém systému lze očekávat delší životnost baterie.

První měření se spouštěnou aplikaci proběhlo v čase mezí 18:15 a 20:15. Snažil jsem se maximálně dodržovat stejné podmínky pro měření a zaříjení byla umístěna v jedné poloze a na jednom místě maximálně blízko od sebe. Po dvou hodinách měření byly odečteny následující hodnoty:

Tabulka 8. Naměřené hodnoty stavu baterie při spuštěné aplikace

Název přístroje	Hodnoty na začátku měření		Hodnoty na konce měření	
	Stav úrovně baterie [%]	Napětí baterie [mV]	Stav úrovně baterie [%]	Napětí baterie [mV]
Sony Xperia Z3 Compact	100	4277	87	4089
HTV One V	100	4166	45	3785
Asus Nexus 7 2013	100	4272	74	3989

Zjištěný stav úrovně baterie má následující průběh (detailnější grafy a průběhy napětí baterie jsou v přílohách Obrázek B. 1, Obrázek B. 2, Obrázek B. 3 a Obrázek B. 4):



Obrázek 23. Průběhy stavu úrovně baterie při měření se spuštěnou aplikace

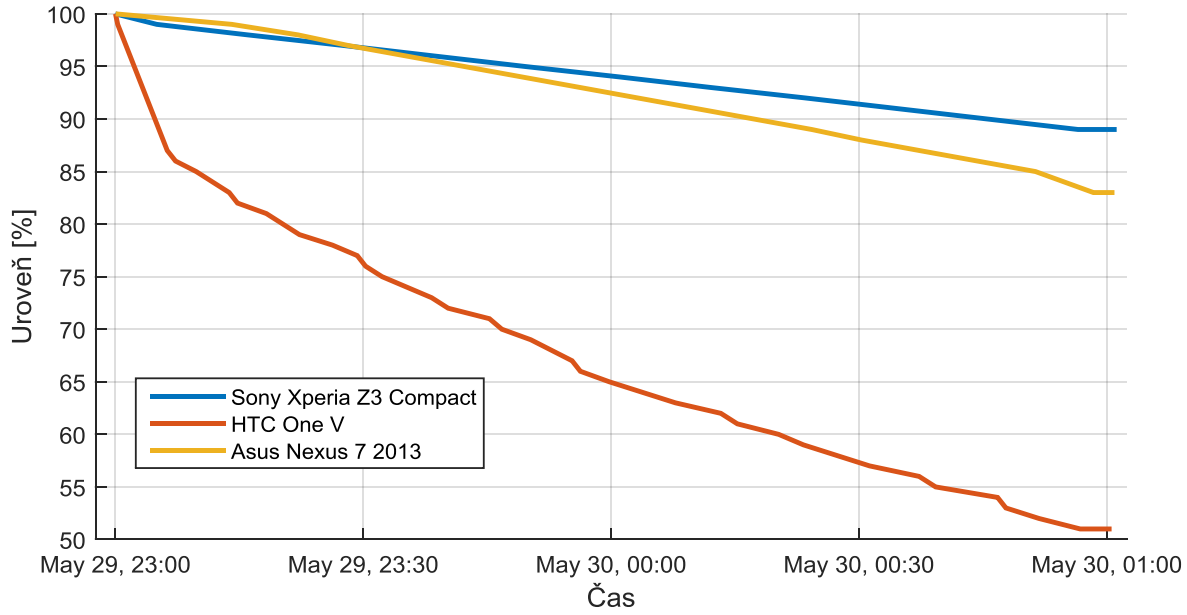
Zdroj: autor

Do jisté míry výsledky pro mě byly překvapující. Zaprvé jsem nečekal, že HTC One V prokáže tak velký pokles napětí. Předpokládal jsem, že kvůli slabší výpočetní jednotce a menší obrazovce, jak rozlišením, tak i velikosti, nároky na vestavenou baterii budou menší. A ze stejného důvodu průběh nejvýkonnějšího přístroje je pro mě neočekávaně mírný. Teď je zapotřebí zjistit, do jaké míry zjištěné průběhy souvisí s aplikací.

Další pokus byl proveden poté, co zařízení byla nabitá a ponechána na další dvě hodiny se zapnutou obrazovkou. Zjištěné hodnoty jsou v Tabulka 9 a průběh stavu baterie je na Obrázek 24.

Tabulka 9. Naměřené hodnoty stavu baterie bez spuštěné aplikace

Název přístroje	Hodnoty na začátku měření		Hodnoty na konce měření	
	Stav úrovně baterie [%]	Napětí baterie [mV]	Stav úrovně baterie [%]	Napětí baterie [mV]
Sony Xperia Z3 Compact	100	4289	89	4160
HTV One V	100	4135	51	3796
Asus Nexus 7 2013	100	4286	83	4053



Obrázek 24. Průběhy stavu úrovně baterie při měření se spuštěnou aplikace

Zdroj: autor

Jak z grafu, tak i z naměřených hodnot je vidět, že aplikace zatěžuje přístroje v malé míře. Je to dáno především v souvislosti s využitím služby zjišťování polohy. A celkově je nutné dbát na to, aby zařízení mělo větší životnost baterie pro časově náročnější měření. Z daného hlediska pokládám aplikaci za využitelnou v praxi, ale se výsledky mohou lišit v závislosti na zařízení, verze operačního systému, a stavu samotné baterie. Proto by bylo vhodné podrobit aplikaci testům na dalších přístrojích.

6.2 Porovnaní s papírovými záznamy

Následujícím kritériem bylo vyhodnocení aplikace z hlediska toho, do jaké míry je schopná nahradit papírové záznamy, porovnat přesnost a časovou náročnost.

Pro porovnání bylo nutné zaznamenávat data, jak do formulářů, tak i do aplikace ve stejné časové okamžiky. Nabízelo se dvě cesty – buď provádět průzkum s asistencí třetí osoby anebo zaznamenat pomoci videokamery a následně data přepsat.

Zvolil jsem druhou možnost, jelikož jsem měl k dispozici podklady z vlastní semestrální prací, která byla vypracována na předmětu Řízení uzlu a linie vyučovaným ústavem dopravní telematiky, a cílem daného testu je jenom porovnání způsobů zaznamenávání dat.

Průzkum intenzit byl proveden v ulici Kbelská, na pěší lávce spojující ulici Kytlická (Praha 9) a Opočenská (Praha 18). Byl počítán jenom celkový počet vozidel, která projela určitým profilem v jednom směru (na jih). Délka trvání byla stanovena na 30 minut.

Ve výsledku pomocí aplikace bylo napočítáno 719 vozidel a v papírových formulářích vyskytlo 715 záznamu. Při průměru 717 vozidel není to tak velká odchylka, která je způsobená především lidským faktorem. Rozdíl je především v tom, jak se záznamy se zacházet po průzkumu.

Při zaznamenávání do papírových formulářů, aby bylo možné získat výsledky je zapotřebí provést následující přepis dat, který je taktéž časově náročný. Když pomoci aplikaci lze zjišťovat hodnoty jak během průzkumu, tak i po skončení během několika vteřin mít data nejenom připravená pro následující zpracování, ale i poskytnuta osobě, která průzkum organizovala.

Ze získaných zkušenosti mohu doporučit nasledujicí. Je vhodné používat se zařízením poutko, aby zabránit škodám, ke kterým může dojit při vyklouznutí přístroje z ruky. Za chladného počasí a použití rukavic je nutné dbát na to, že ne všechny dotykové obrazovky jsou schopné zaznamenávat doteky přes látku. Proto doporučují buď ověřit, jestli zařízení danou funkci podporuje anebo provádět průzkum v rukavicích s elektrovodivými konečky.

6.3 Výhled do budoucna

Pokud by aplikace měla využití v praxi a byl by o ní zájem, nabízejí se další funkce, které lze implementovat.

Za nejdůležitější pokládám nutnost jednodušší úpravy rozhraní pro průzkumy. Vidím to následujícím způsobem: vzhled a parametry rozhrání budou uloženy v souboru, který lze nahrát do pamětí přístroje a následně v aplikaci zvolit možnost načtení vzhledu z příslušného souboru. Tahle možnost by umožnila provádět další druhy průzkumu bez nutností kompilace

nové aplikaci. Dále se nabízí možnost komunikace softwaru s vlastním serverem, na který mohou být nahraný výsledky a sledován průběh průzkumů.

Předchozí dva body lze spojit a vytvořit celý nástroj pro vzdálenou zprávu zařízení, vizuální úpravu rozhraní, nahrávání souboru do zařízení, umožnit komunikaci zařízení mezi sebou atd.

6.4 Shrnutí

V kapitole 2 byly zjištěny požadavky na software včetně funkcí, které aplikace musí nabízet, jak z hlediska organizátora průzkumu, tak i z hlediska uživatele. Dále uvedu seznam požadavku ze zmíněné kapitoly a podíváme se, na to, co se splnit podařilo.

- Aplikace musí nahradit papírové záznamy: ve větší míře aplikace je schopná nahradit papírové záznamy. Výjimku mohou tvořit velmi zvláštní průzkumy, například, za chladného počasí, kdy může dojít k poklesu kapacity baterie.
- Vynechání přepis dat ze záznamového přístroje do elektronické podoby: podařilo se realizovat v plné míře.
- Uložená data musí mít přehlednou strukturu: díky použití XML formátování struktura souboru je přehledná i pro nezkušeného uživatele.
- Realizace alespoň jednoho univerzálního rozhraní pro průzkumy intenzit: čtrnáct kategorií vozidel v dostatečné míře splňují požadavky na kategorizaci vozidel.
- Možnost přizpůsobení rozhrání: podařilo se realizovat jenom částečně. Bohužel pro změnu ikon na obrazovce pro průzkumy je zapotřebí aplikaci znovu zkompilovat.
- Zabezpečení a zálohovaní dat: zálohování výsledků měření probíhá během exportu. Databáze jsou uložené na interní paměti zařízení, výstupní soubor na externí a může být přeposlán uživatelem na vzdálené uložiště. Data v databázi nejsou přístupna uživateli a nemohou být padělána.
- Oprava chybných záznamů: je realizovaná v plné míře.
- Zaznamenávání polohy stanoviště, které je vhodné pro kontrolu průzkumu, organizátorem je taktéž realizováno v plném rozsahu.
- Synchronizace času kvůli omezení operačního systému není realizováno přímo v aplikaci, ale samotný operační systém umí sám synchronizovat hodiny bez zásahu uživatele..

7 Závěr

Nehledě na to, že v současné době díky technickému pokroku nabízené množství dopravních detektorů umožňuje snadné získávání charakteristik dopravního proudů za všelijakých podmínek, stále jsou situace, kde jejich použití je omezeno různými faktory. Patří k nim pořizovací náklady nebo neúčelnost při celoplošných a krátkodobých průzkumech. Ale nezbytnost měřit dopravu v podobných případech nutí k použití zastaralých a náročných způsobu zaznamenávaní dat. Právě v dané oblasti je prostor pro realizaci nápadů, které by usnadnily proces zápisu dat a nabídly nové možnosti zpracování.

Cílem této diplomové prací bylo navrhnout software, který by, alespoň v nějaké míře, usnadnil dopravní průzkumy, a aby práce se záznamy neodřazovala nezkušené studenty naší fakulty. První část práce je věnovaná analýze požadavku na software, kde byly zjištěny a stanoveny parametry dopravních proudu, které je možné získávat prostřednictvím dopravních průzkumů. Stanoveny problematické způsoby měření, které je vhodné optimalizovat podporným softwarem a vyřešen způsob ukládání dat, poskytující snadné zpracování záznamů a jejích zabezpečení. Před návrhem proběhlo vyhodnocení platforem poskytujících dostatečný prostor pro realizaci vlastního nápadu v souladu s požadavky. Pro zvolený operační systém byla zjištěna vývojová prostředí, ve kterých bylo možné software vyvíjet, uskutečněno jejích porovnání z hlediska nabízených možností a finančních nákladů. Realizace softwaru se skládala z návrhu uživatelského rozhrání, vyřešení způsobu ukládání dat a samotného vývoje. V závěrečné části byla ověřena funkčnost aplikaci a byly nastíněny funkce pro implementaci v budoucnu.

Realizovaný software byl navrhnout pro podporu ručních měření a nahrazení papírových záznamů, především z důvodu časové náročnosti přepisu dat z papírových formulářů do elektronické podoby. Aplikace byla vyvinuta pro operační systém Android díky jeho rozšíření a cenové dostupnosti. Ze stejných důvodů vývoj probíhal ve standardním vývojovém prostředí Android Studio dodávaném společností Google. Funkčnost byla ověřena na zařízeních s různými verzemi operačního systému a bylo zjištěno, že přístroje s operačním systémem Android umožnují prostřednictvím realizovaného softwaru v dostatečné míře nahradit papírové záznamy.

Součásti elektronických příloh jsou zdrojové kódy realizované aplikace, spustitelný soubor APK pro instalací softwaru do zařízení s operačním systémem Android 4.2+.

Tištěné přílohy obsahují grafy vyskytující v textu kapitoly 6.1, a které jsou zobrazeny ve zvětšeném měřítku.

Textová část práce byla zpracována v programu MS Word, pro vykreslení grafu byl použit program MATLAB, zvukové soubory byly vytvořeny pomocí otevřeného softwaru Audacity, ikony použité v aplikaci byly vytvořeny autory Freepik a OCHA z webu www.flaticon.com a spadají pod licenci Creative Commons BY 3.0. Návrh databáze se prováděl pomocí nástrojů WWW SQL Designer.

Věřím, že tyto poznatky a navržená řešení budou použity v praxi.

8 Použitá literatura

[1] ČESKÝ STATISTICKÝ ÚŘAD. Dokončené byty v krajích (ročně) . *Veřejná databáze ČSÚ* [online]. 2015 [cit. 2015]. Dostupné také z: https://vdb.czso.cz/vdbvo/tabparam.jsp?voa=tabulka&cislotab=BYT0030PU_KR&vo=tabulka&kapitola_id=35

[2] BARTOŠ, Luděk. *Stanovení intenzit dopravy na pozemních komunikacích*. 2. vyd. Plzeň: EDIP, 2012. ISBN: 978-80-87394-06-9.

[3] KŘIVDA, Vladislav. *Problematika homogenizace dopravního proudu v silnční dopravě* [online]. Číslo 3. Pardubice: Univerzita Pardubice, Dopravní fakulta Jana Pernera, Katedra technologie a řízení dopravy, 2008, s. 39-43 [cit. 2015]. ISSN: 1801-674X.

[4] ELEFTERIADOU, L.. *An Introduction to Traffic Flow Theory*. Springer, 2013. ISBN: 9781461484356. Dostupné také z: https://books.google.cz/books?id=VOK5BAAAQBAJ

[5] TRANSPORTATION RESEARCH BOARD, NATIONAL RESEARCH COUNCIL. *Highway capacity manual*. Washington, DC: Transportation Research Board, National Research Council, 2000.

[6] ŘEDITELSTVÍ SILNIC A DÁLNIC ČR. Sčítání dopravy a detekce kolon. *Dopravní info* [online]. 2009 [cit. 2015]. Dostupné také z: http://www.dopravniinfo.cz/scitani-dopravy-a-detekce-kolon

[7] RICHARDSON, A.J., E.S. AMPT a A.H. MEYBURG. *Survey Methods for Transport Planning*. Eucalyptus Press, 1995. ISBN: 9780646214399. Dostupné také z: https://books.google.cz/books?id=bwHjAAAACAAJ

[8] LEDUC, Guillaume. Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change*. 2008, **1**: 55.

[9] REŽŇÁK, Tomáš. *Průběh sčítání CSD* [online]. 2010 [cit. 2015]. Dostupné také z: http://csd.cdv.cz/prubeh-scitani-csd/

[10] BRAY, Tim, Jean PAOLI, C SPERBERG-MCQUEEN, Eve MALER a Fran{\c{c}}ois YERGEAU. Extensible markup language (XML). *World Wide Web Consortium*

Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210. 1998, : 16.

[11] SHAFRANOVICH, Yakov. *Common format and MIME type for Comma-Separated Values (CSV) files*. 2005.

[12] BEN-KIKI, Oren, Clark EVANS a Brian INGERSON. YAML Ain't Markup Language (YAMLâ„˘) Version 1.1. *Yaml. org, Tech. Rep*. 2005.

[13] NURSEITOV, Nurzhan, Michael PAULSON, Randall REYNOLDS a Clemente IZURIETA. Comparison of JSON and XML Data Interchange Formats: A Case Study. *Caine*. 2009, **2009**: 157-162.

[14] DAMOULAKIS, J. Continuous protection. *Storage, June 2004*. 2004, **3**(4): 33-39.

[15] RECORDABLE, Disc. *ECMA-377*. b.r..

[16] PATTERSON, David, Garth GIBSON a Randy KATZ. A case for redundant arrays of inexpensive disks (RAID). *ACM Sigmod Record*. ACM, 1988, **17**(3): 109-116.

[17] KPMG INTERNATIONAL COOPERATIVE. *Data Loss Barometer* [online]. Londýn: RR Donnelley, 2012 [cit. 2015]. Dostupné také z: https://www.kpmg.com/EE/et/IssuesAndInsights/ArticlesPublications/Documents/Data-Loss-Barometer.pdf

[18] ČERNÝ, Aleš. SERIÁL: Liftago chce ušetřit "statisíce zbytečně najetých" kilometrů pražských taxikářů. *Ihned.cz* [online]. Praha, 2014 [cit. 2015]. Dostupné také z: http://byznys.ihned.cz/analyzy-a-komentare/c1-62508770-serial-liftago-chce-usetrit-statisice-zbytecne-najetych-kilometru-prazskych-taxikaru

[19] STATISTA, . *Mobile operating systems: global market share 2012-2015 Statistic*. 2015. Dostupné také z: http://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[20] STATISTA, . *Smartphones sales by operating system worldwide 2009-2014 Statistic*. 2015. Dostupné také z: http://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/

[21] STATT, Nick. *Microsoft Windows Phone by any other name: It's now 'Windows 10 Mobile'*. 2015. Dostupné také z: http://www.cnet.com/news/microsoft-drops-windows-phone-in-favor-of-windows-10-mobile/

[22] MOTION, Research. *BlackBerry 10 - BlackBerry Q5, BlackBerry Q10 \& BlackBerry Z10*. b.r.. Dostupné také z: http://global.blackberry.com/blackberry-10.html

[23] SAMSUNG, . *Bada Developers*. b.r.. Dostupné také z: http://developer.bada.com

[24] PROJECT, Tizen a Linux PROJECT. *An open source, standards-based software platform for multiple device categories.* b.r. Dostupné také z: https://www.tizen.org/

[25] NETWORK, Mozilla. *Firefox OS*. b.r.. Dostupné také z: https://developer.mozilla.org/en-US/Firefox_OS

[26] SHAIKH, Asif. *Performance Analysis: Ubuntu Touch v/s Android OS*. b.r..

[27] ZEMAN, Eric. *Microsoft's Windows Phone 8: The Bad*. 2012. Dostupné také z: http://www.informationweek.com/software/operating-systems/microsofts-windows-phone-8-the-bad/d/d-id/1104950?

[28] GOOGLE INC. *Android Developers* [online]. 2015 [cit. 2015]. Dostupné také z: http://developer.android.com

[29] WOLBER, David , Hal ABELSON, Ellen SPERTUS a Liz LOONEY. *App Inventor*. Sebastopol, Calif.: O'Reilly Media, Inc., 2011. 9781449397487.

[30] LABS, Corona. *Corona SDK*. b.r.. Dostupné také z: https://coronalabs.com/products/corona-sdk/

[31] HIPBYTE, . *HipByte/RubyMotionSamples*. b.r.. Dostupné také z: https://github.com/HipByte/RubyMotionSamples/tree/master/android/Hello

[32] KIVY. *Kivy: Cross-platform Python Framework for NUI* [online]. 2015 [cit. 2015]. Dostupné také z: http://kivy.org/#home

[33] SMITH, HM. The Greenwich Time Signal Service. *Annales frangaises de chronometrie*. 1949, : 227.

[34] INC., Sony. *Xperia Z3 Compact*. b.r.. Dostupné také z: http://www.sonymobile.com/global-en/products/phones/xperia-z3-compact/specifications/\#tabs

[35] GOOGLE, . *Nexus 7 (2013) Tech Specs*. b.r.. Dostupné také z: https://support.google.com/nexus/answer/3247662?hl=en

[36] CORPORATION, HTC. *HTC One V Specifikace a recenze*. b.r.. Dostupné také z: http://www.htc.com/cz/smartphones/htc-one-v/\#/

[37] BRIGHT, Peter. *Windows NT coming to phones with Windows Phone 8*. b.r.. Dostupné také z: http://arstechnica.com/gadgets/2012/06/windows-nt-coming-to-phones-with-windows-phone-8/

[38] ALLSOPP, Clay. *RubyMotion*. United States: The Pragmatic Programmers, 2013. ISBN: 9781937785284.

[39] *WWW SQL Designer*. b.r.. Dostupné také z: http://ondras.zarovi.cz/sql/demo/

[40] *GPS.gov: Timing Applications*. b.r.. Dostupné také z: http://www.gps.gov/applications/timing/

[41] Леонидович, Г.А.. *Google Android: программирование для мобильных устройств (+ CD) - 2-е издание*. БХВ-Петербург, 2012. ISBN: 9785977507295. Dostupné také z: https://books.google.cz/books?id=hx36iz5za54C

[42] TICHÝ, Tomáš. *Řídicí systémy dopravy - dopravní telematika*. Praha, 2004. unpublished.

[43] SHAPIRO, Jeffrey a Jim BOYCE. *Windows 2000 server bible*. Wiley, 2000.

[44] ROESS, R.P., E.S. PRASSAS a W.R. MCSHANE. *Traffic Engineering*. Pearson/Prentice Hall, 2004. ISBN: 9780131918771 LCCN: 89023159. Dostupné také z: https://books.google.cz/books?id=akMaQAAACAAJ

[45] MAY, A.D.. *Traffic flow fundamentals*. Prentice Hall, 1990. LCCN: 89039687. Dostupné také z: https://books.google.cz/books?id=XikpAQAAMAAJ

[46] KLEIN, Lawrence, Milton MILLS a David GIBSON. *Traffic Detector Handbook: - Volume II*. 2006. Tech. rep.

[47] KERNER, B.S.. *Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory*. Springer Berlin Heidelberg, 2009. ISBN: 9783642026058. Dostupné také z: https://books.google.cz/books?id=4g7f1h4BfYsC

[48] KELL, James, Iris FULLERTON a Milton MILLS. *Traffic detector handbook*. 1990. Tech. rep.

[49] HARDY, B. a B. PHILLIPS. *Android Programming: The Big Nerd Ranch Guide*. Pearson Education, 2013. ISBN: 9780132869102. Dostupné také z: https://books.google.cz/books?id=OFXJXbCXjTgC

[50] CROCKFORD, Douglas. Introducing json. *Available: json. org*. 2009.

[51] BRAY, Tim. *The JavaScript Object Notation (JSON) Data Interchange Format*. 2014.

[52] (ORGANIZATION), Standards a Standards ZEALAND. *Information Technology: Security Techniques Code Of Practice For Information Security Management*. Wellington N.Z.: Standards New Zealand, 2006. ISBN: 9780733774928. Dostupné také z: http://isbnplus.org/9780733774928

9 Seznam obrázků

10 Seznam tabulek

11 Seznám zdrojových kódů

12 Seznam příloh

12.1 Příloha A

12.2 Příloha B

12.3 Příloha C

Zdrojové kódy realizovaného softwaru

12.4 Příloha D

Spustitelný soubor pro instalaci aplikace do zařízení

Přílohy

Příloha A. Ukázky zdrojových kódů, použitých v aplikaci.

Soubor: MainActivity,java (řádky 148-176)

```
btnIntenzity = (Button) findViewById(R.id.btnItenzity);
View.OnClickListener oclBtnIntenzity = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // connecting to DB
        SQLiteDatabase dbPruzkumy = dbHelper.getWritableDatabase();
        // creating object for data
        ContentValues cvPruzkum = new ContentValues();
        // prepare data
        calBegin.set(Calendar.SECOND, 0);
        calEnd.set(Calendar.SECOND, 0);
        cvPruzkum.put(ZACATEK, calBegin.getTimeInMillis());
        cvPruzkum.put(KONEC, calEnd.getTimeInMillis());
        cvPruzkum.put(INTENZITA, 0);
        cvPruzkum.put(SOURADNICE, "N/A");
        // insert data
        Log.d(TAG, "data prepared");
        long rowID = dbPruzkumy.insert("pruzkumy", null, cvPruzkum);
        Log.d(TAG, "data inserted with id = " + rowID);
        dbHelper.close();
        Intent intentPruzkum = new Intent(MainActivity.this,
MainActivityIntenzity.class);
        // send extra data to new activity
        intentPruzkum.putExtra("begin", (Long) calBegin.getTimeInMillis());
        intentPruzkum.putExtra("end", (Long) calEnd.getTimeInMillis());
        intentPruzkum.putExtra("id_pruzkumu", (Long) rowID);
        Log.d(TAG, "opening other activity" + rowID);
        startActivity(intentPruzkum);
    }

};
```

Zdrojový kód A. 1. Ukázka kódu, který je vykonán při doteku tlačítka pro začátek nového průzkumu

Soubor: MainActivity,java (řádky 185-231)

```
View.OnClickListener oclBtnSmazat = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //create dialog warning
        new AlertDialog.Builder(MainActivity.this)
                .setIcon(R.drawable.ic_action_alert)
                .setTitle(R.string.deleteAlert)
                .setMessage(R.string.deleteAlertMsg)
                .setPositiveButton(R.string.sBtnSmazat,
                        new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which){
                        // connecting to DB
                        pos = spinnerVyberPruzkumu
                                .getSelectedItemPosition();
                        deleteXML(pos);
                        SQLiteDatabase dbPruzkumy =
                                dbHelper.getWritableDatabase();
                        Log.d(TAG, "Pos list view = " + pos);
                        Cursor cursor = dbPruzkumy.query(
                                TABLE_PRUZKUMY,
                                null, null, null, null, null, null);
                        cursor.moveToPosition(pos);
                        //rewrite value of POS to id from table
                        pos = cursor.getInt(cursor.getColumnIndex(ID));
                        String sPos = "" + pos;
                        dbPruzkumy.delete(TABLE_PRUZKUMY,
                                ID + " = " + pos, null);
                        int del = dbPruzkumy.delete(
                                TABLE_ZAZNAMY,
                                ID_PRUZKUMU + " = " + pos, null);
                        Log.d(TAG,
                                "delete row with" + ID +" = " + pos);
                        Log.d(TAG,
                                "delete " + del + " records from zaznamy");
                        Toast.makeText(MainActivity.this,
                                R.string.deleteOk,
                                Toast.LENGTH_SHORT).show();
                        cursor.close();
                        db2spinner();
                        dbHelper.close();
                    }
                })
                .setNegativeButton(android.R.string.cancel, null)
                .show();
    }
};
```

Zdrojový kód A. 2. Ukázka kódu, který je vykonán při doteku tlačítka pro odstranění záznamů

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Průzkum>
    <Vlastnosti>
        <Lokalita>
            <souřadnice>N/A</souřadnice>
        </Lokalita>
        <Začátek>
            <datum>2015-05-28</datum>
            <čas>15:17:00</čas>
            <pásmo>SEČ</pásmo>
        </Začátek>
    </Vlastnosti>
    <Záznamy>
        <událost>
            <Vozidlo>Osobní</Vozidlo>
            <Datum>2015-05-28</Datum>
            <Čas>15:17:47.733</Čas>
        </událost>
        <událost>
            <Vozidlo>Bus kloub.</Vozidlo>
            <Datum>2015-05-28</Datum>
            <Čas>15:17:47.955</Čas>
        </událost>
        <událost>
            <Vozidlo>Bus solo</Vozidlo>
            <Datum>2015-05-28</Datum>
            <Čas>15:17:48.159</Čas>
        </událost>
        <událost>
            <Vozidlo>Moto</Vozidlo>
            <Datum>2015-05-28</Datum>
            <Čas>15:17:48.365</Čas>
        </událost>
        <událost>
            <Vozidlo>Osobní</Vozidlo>
            <Datum>2015-05-28</Datum>
            <Čas>15:17:48.503</Čas>
        </událost>
    </Záznamy>
</Průzkum>
```

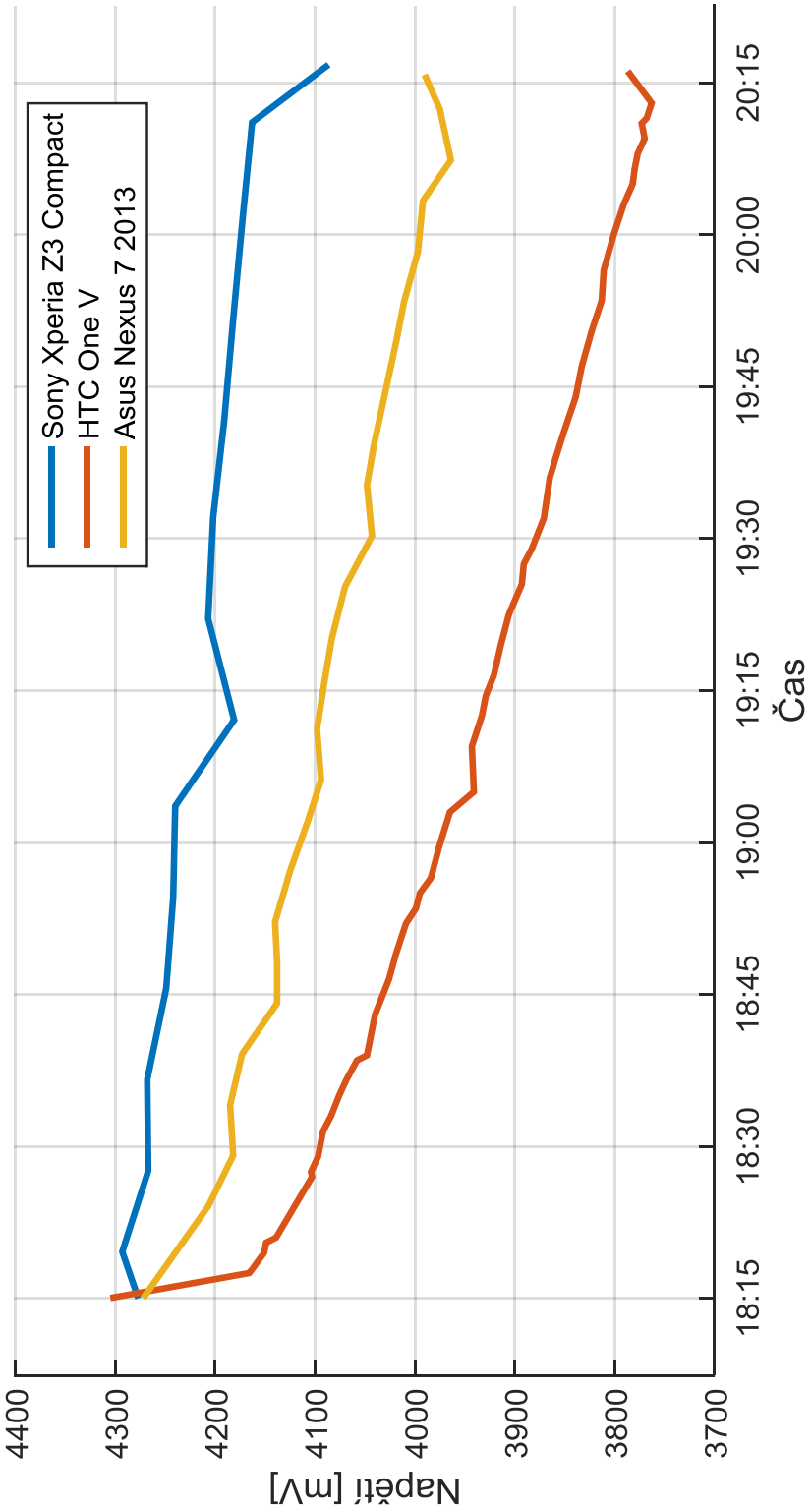
Zdrojový kód A. 3. Ukázka XLM souboru se záznamy, který je vygenerován aplikací

Soubor: MainActivityIntenzity.java (řádky 137-183)

```
private LocationListener locationListener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        dLat = location.getLatitude();
        dLong = location.getLongitude();
        tvGps.setText("" + dLat + ", " + dLong +
                " (\u00B1" +
                Float.toString(location.getAccuracy()) + ")");
        tvGps.setTextColor(getResources().
                getColor(android.R.color.holo_green_dark));
    }

    @Override
    public void onStatusChanged(String provider,
                                int status, Bundle extras) {
        switch (status) {
            case 0: //OUT_OF_SERVICE
                tvGps.setText("N/A");
                tvGps.setTextColor(getResources().
                        getColor(android.R.color.holo_red_dark));
                break;
            case 1: //TEMPORARILY_UNAVAILABLE
                //tvGps.setText("TEMPORARILY UNAVAILABLE");
                tvGps.setTextColor(getResources().
                        getColor(android.R.color.holo_orange_dark));
                locationManager.requestSingleUpdate(
                        LocationManager.GPS_PROVIDER,
                        locationListener, null);
                break;
            case 2:
                tvGps.setTextColor(getResources().
                        getColor(android.R.color.holo_green_dark));
                locationManager.requestSingleUpdate(
                        LocationManager.GPS_PROVIDER,
                        locationListener, null);
                break;
        }
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }
};
```

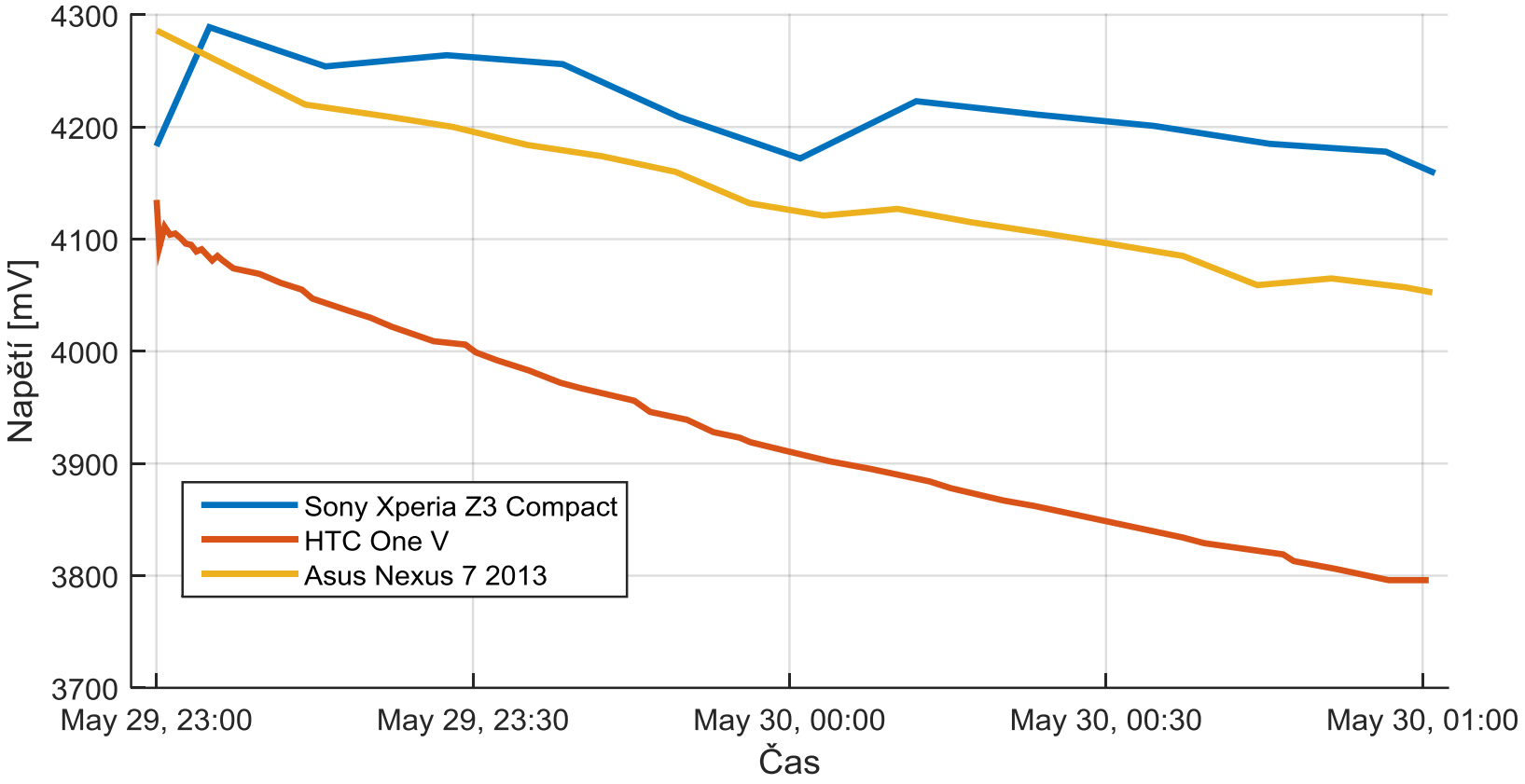
Zdrojový kód A. 4. Realizace objektu, poskytujícího informace o aktuální poloze

Příloha B. Grafy stavů baterie.



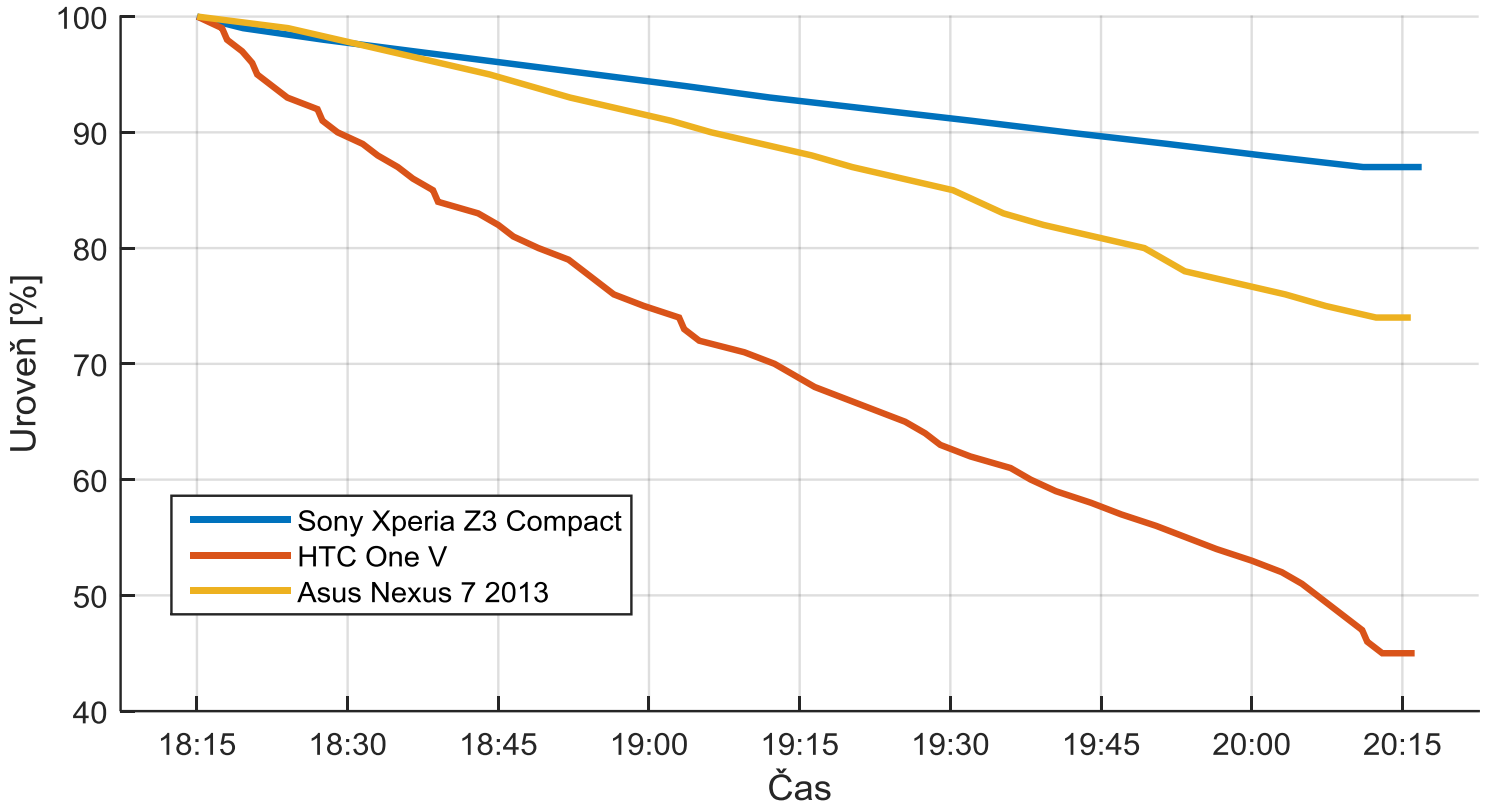
Obrázek B. 1. Průběhy stavu napětí při měření se spuštěnou aplikací

Zdroj: autor



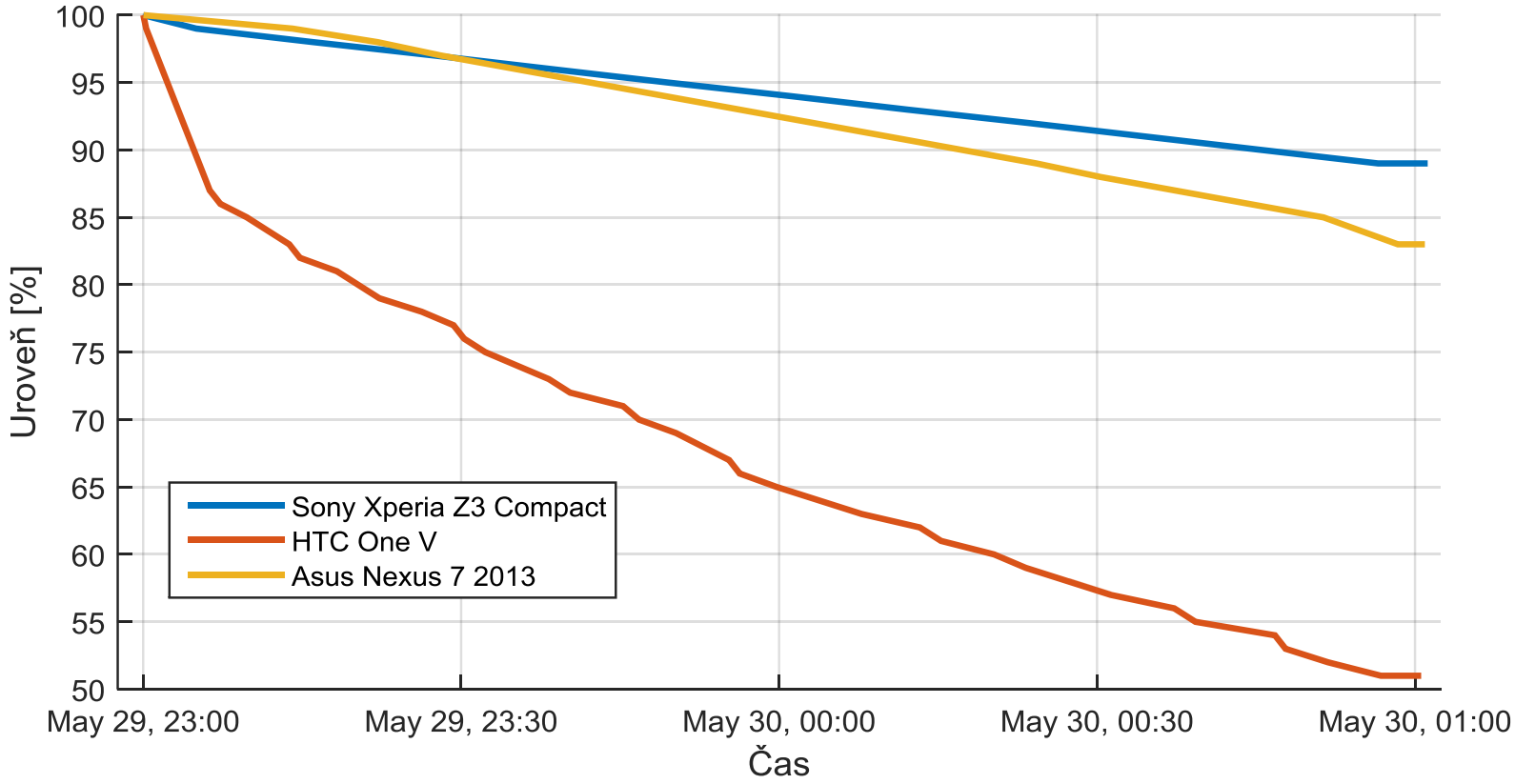
Obrázek B. 2. Průběhy stavu napětí při měření bez spuštěné aplikace

Zdroj: autor



Obrázek B. 3. Průběhy stavu úrovně baterie při měření se spuštěnou aplikací

Zdroj: autor



Obrázek B. 4. Průběhy stavu úrovně baterie při měření bez spuštěné aplikace

Zdroj: autor