Vývoj multiplatformní aplikace pro sledování polohy vozidel městské hromadné dopravy

Bc. Michal Paukert

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal Paukert**
Osobní číslo:	**A15441**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Vývoj multiplatfomní aplikace pro sledování polohy vozidel městské hromadné dopravy**
Téma anglicky:	**The Development of a Multi-platform Application for the Tracking of Public Transport Vehicles**

Zásady pro vypracování:

1. **Vypracujte rešerši na dané téma**
2. **Analyzujte výběr možných technologií**
3. **Navrhněte serverou aplikaci jako agregátor dat z předem daných zdrojů**
4. **Vytvořte mobilní aplikaci pro zobrazování dat z API**
5. **Aplikaci otestujte na reálném zařízení**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Dan Hermes.Xamarin Mobile Application Development: Cross-Platform Cand Xamarin.Forms Fundamentals. Apress, 2015.**
2. **Jonathan Peppers.Xamarin Cross-platform Application Development – Sec-ond Edition. Packt Publishing, 2015.**
3. **Mark Reynolds.Xamarin Mobile Application Development for Android.Packt Publishing, 2014.**
4. **Matthew Leibowitz.Xamarin Mobile Development for Android Cookbook.Packt Publishing, 2015.**
5. **Scott Olson, John Hunter, Ben Horgen, and Kenny Goers.ProfessionalCross-Platform Mobile Development in C. Wrox, 2012.**
6. **Charles Petzold.Creating Mobile Apps with Xamarin.Forms Preview Edition2. 2015.**
7. **Developer Center – Xamarin [online]. Xamarin HQ: Xamarin, 2016 [cit. 2017-02-02]. Dostupné z: https://developer.xamarin.com/**

Vedoucí diplomové práce: **Ing. Bc. Pavel Vařacha, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně
12.5.2017

.....
podpis autora

ABSTRAKT

Diplomová práce se zabývá tvorbou multiplatformní aplikace zaměřené na zobrazování polohy vozidel městské hromadné dopravy. Aplikace zobrazuje vozidla na mapě, informace o nich a jejich směru. Teoretická část má za cíl porovnat výhody a nevýhody technologií pro vývoj multiplatformních mobilních aplikací a vybrat tu nejvhodnější z nich pro vývoj této aplikace. Další částí je vývoj serverové aplikace, která má za úkol fungovat jako agregátor dat ze systému poskytujících aktuální data o městské hromadné dopravě tak, aby nedošlo k přetěžování celého systému.

Klíčová slova: multiplatformní vývoj, mobilní aplikace, xamarin, mhd

ABSTRACT

Thesis deals with the development of a multiplatform application focused at displaying the position of public transport vehicles. Application displays the vehicles on the map, information about them and their direction. The theoretical part is comparing the advantages and disadvantages of technology for the development of multiplatform mobile applications and choose the most suitable for the development of this application. Another part is the development of a server application that is intended as a data aggregator from the third party systems providing up-to-date data of urban public transport and do not overload the entire system

Keywords: multiplatform development, mobile application, xamarin, public transport

Rád bych poděkoval Ing. Bc. Pavel Vařachovi, Ph.D. za vedení diplomové práce a čas strávený při konzultacích. Rovněž bych rád poděkoval všem, kteří mi pomáhali zejména s testováním aplikace, kontrolou pravopisu a textu.

OBSAH

ÚVOD

Chytré telefony se stávají neodmyslitelnou součástí našeho života. Za posledních pár let se tak staly nejrozšířenější informační technologií v českých domácnostech. Díky jejich chytrým funkcím a rozvoji mobilního datového připojení přibývá uživatelů na internetu. Toto tvrzení vyplývá z posledních dat ČSÚ, kdy na sto domácností připadá 206 uživatelů. Největší část této populace tvoří jednotlivci starší 16ti let. [2]

Chytré telefony nám mohou velice zefektivnit náš čas, v případě že budeme mít dostatek informací a zároveň budou správně reprezentovány. Například v dopravě, pokud budeme mít přístup k informacím o tom, že na námi vybrané trase dochází k dopravní koloně, můžeme se jí vyvarovat a zvolit tak jinou trasu nebo věci přeorganizovat na jiný čas. Takovouto informaci již jsme schopni získat z mnoha aplikací jako je například Waze, Google Maps a jiné. Toto však platí zejména pro cestování autem, ale v případě cestování městskou hromadnou dopravou to nemusí platit. Zejména pokud se jedná o regionální vlakovou dopravu nebo v případě tramvají, pro které jsou mnohdy vyčleněny tramvajové pásy, aby nemusely stát v koloně aut a byly tak rychlejší.

Tento problém pro určitou skupinu lidí řeší mobilní aplikace jako je například Wifi Tram, iRIS a Šotoris, které umožňují vizualizaci spojů na mapě. Nicméně aplikace iRIS je vytvořena pouze pro platformu Android a Šotoris momentálně pouze pro Windows Phone a BlackBerry. Navíc obě aplikace spojuje omezení stejného poskytovatele dat. Zobrazují tedy data pouze z Jihomoravského kraje. Uživatelé platformy iOS jsou momentálně odkázáni na aplikaci Easy Brno, která není primárně určena k zobrazování spojů na mapě, ale má i tuto funkcionalitu. Další takovouto aplikaci je IDOS, která sice nenahrazuje zmíněné předešlé aplikace, ale informuje uživatele alespoň o zpoždění spoje a taktéž pouze v rámci Jihomoravského kraje. Je velkou škodou, že není příliš mnoho měst, která by poskytovala data o spojích. Pokud tato data existují a jsou dostupná, většinou jsou zobrazována pouze ve formě webové aplikace a mobilní aplikace k nim chybí. Mezi nejznámější města disponující těmito informacemi, nebo alespoň ve webové formě, jsou Praha (pouze Tram), Brno a Zlín.

V teoretické části této práce bude čtenář seznámen s jednotlivými mobilními platformami a použitými technologiemi. Popis návrhu jak mobilní aplikace tak i serverového backendu včetně popisu získávání dat bude popsán v analytické části. Praktická část má popsat vlastní implementaci a zobrazit výstup této práce.

Hlavním cílem této práce je tedy vyvinout multiplatformní mobilní aplikaci zaměřenou na hlavní mobilní platformy. Dalším z cílů je odstranění omezení pouze na jednoho poskytovatele dat a následné využití těchto dat v mobilní aplikaci.

I. TEORETICKÁ ČÁST

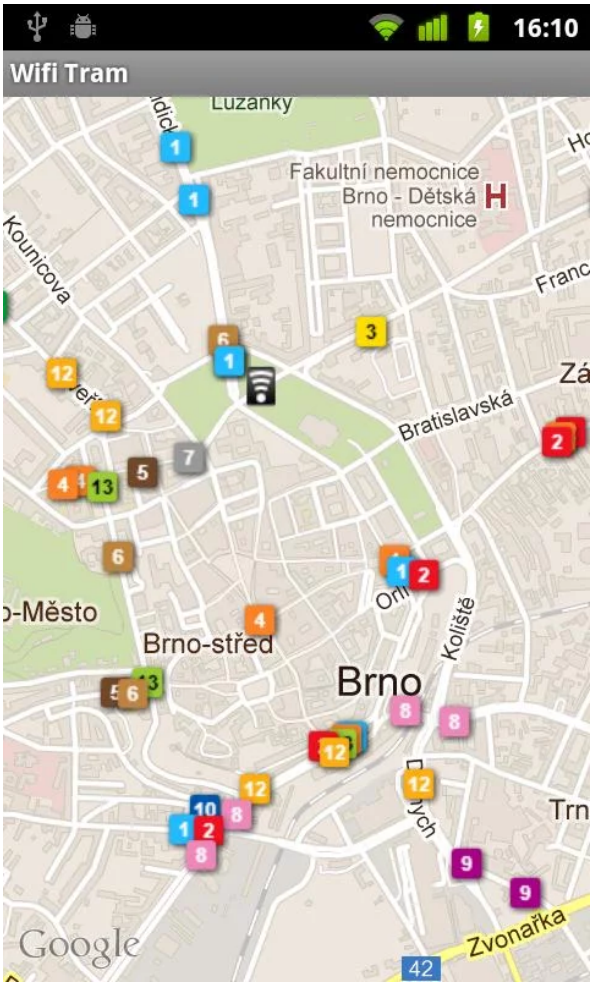
1 Analýza současných možností

Jak již bylo řečeno v úvodu, uživatel je velmi limitován typem mobilní platformy. S jistotou můžeme říct, že uživatelé platformy Android jsou momentálně v největší výhodě. I přesto, že neexistuje příliš mnoho mobilních aplikací, které by nám umožnily zobrazovat spoje přímo na mapě, právě pro Android jich existuje nejvíce. V Google Play můžeme tedy najít aplikace Wifi Tram, iRIS, Easy Brno a jiné. Naproti tomu platforma Windows Phone je na tom velmi bídně a nejen v tomto případě. Chybí alternativy pro aplikace celkově. Microsoft Store disponuje aplikací podobného typu pouze aplikací Šotoris a Easy Brno. Platforma iOS je na tom co do počtu aplikací o něco lépe než Windows Phone, nicméně v případě aplikací tohoto typu je na tom podobně jako Windows Phone. Jediná funkční nativní aplikace, která se dá stáhnout z App Store je Easy Brno, ta však není primárně určena pouze ke sledování polohy MHD. Do nedávna tomu tak nebylo a byla k dispozici aplikace Šotoris, ta však byla stažena z App Store.

1.1 Wifi Tram

„WifiTram je aplikace, která zobrazuje okamžitou polohu tramvají, trolejbusů, autobusů i lodí Dopravního podniku města Brna. Data o poloze vozidel jsou přenášena v reálném čase přímo z dispečinku systému RIS DPMB.“ [23] Vznikla v roce 2011 v rámci reklamní kampaně, kde některá vozidla MHD byla vybavena Wi-Fi hotspoty. Současně má i webovou formu [24]. Aplikace je sice funkční, nicméně je již dlouho dobu neaktualizována a ztrácí tak na atraktivnosti. To potvrzuje i celkové hodnocení v obchodě Google Play. Její aktuální podoba je zobrazena na obrázku 1.1

- Platforma: Android, Web
- Lokace dat: Brno a okolí
- Datum vzniku: 2011
- Google Play [23]



Obr. 1.1 Aplikace Wifi Tram [23]

1.2 iRIS

„Aplikace zobrazuje aktuální polohu všech vozidel v systému IDS JMK, včetně MHD Brno, Blansko, Kyjov, Břeclav, Vyškov, Adamov, Hodonín, Znojmo, regionálních autobusů, vlaků a další informace o vozidlech. Poloha vozidel se přenáší pomocí dopravních dispečinků CEDRIS a RIS. Poloha vozidel je aktualizována přibližně každých 20 sekund.“ Momentálně můžeme tuto aplikaci považovat za jednu z nejlepších na českém trhu. Je často aktualizována, má velké množství funkcí a tedy i velmi dobré hodnocení. Její webovou verzi je možné nalézt na adrese [26], mobilní aplikace je pak zobrazena na obrázku 1.2 [25]

- Platforma: Android, Web
- Lokace dat: Jihomoravský kraj
- Datum vzniku: 2012

- Google Play [25]

Obr. 1.2 Aplikace iRIS [25]

1.3 Šotoris

„Aplikace zobrazuje na mapě aktuální polohu vozidel Dopravního podniku města Brna. Zdrojová data o poloze tramvají, autobusů a trolejbusů v dopravní síti města Brna poskytuje Dopravní podnik města Brna, a. s.“. [27] [28] Tato aplikace se uchytila zejména díky dostupnosti na jiných platformách než je Android. Momentálně to jsou veškeré mobilní zařízení Windows a Blackberry. Nevýhoda této aplikace oproti předchozím je, že nezobrazuje celý Jihomoravský kraj, ale pouze okolí města Brna. Náhled aplikace je prezentován na obrázku 1.3

- Platforma: Windows Phone, Blackberry, dříve i iOS
- Lokace dat: Brno a okolí

- Datum vzniku: 2011
- Microsoft Store [27]

Obr. 1.3 Mobilní aplikace Šotoris [27]

1.4 Easy Brno

„Aplikace Easy Brno (dříve Modré Brno) je pomocník v mobilu, který vám v Brně najde vše, co právě potřebujete. Velmi oblíbené je například zobrazení aktuální polohy MHD v Brně.“ [29] Tato aplikace je určena pro platformy iOS, Android i Windows Phone. Je velmi zajímavá v tom, že poskytuje komplexní informace, které by mohly občany města Brna zajímat, nicméně část zobrazující polohu MHD působí příliš "přeplácaně". [29] Podobu této aplikace lze vidět na obrázku 1.4

- Platforma: Android, iOS, Windows Phone
- Lokace dat: Brno a okolí
- Datum vzniku: 2015

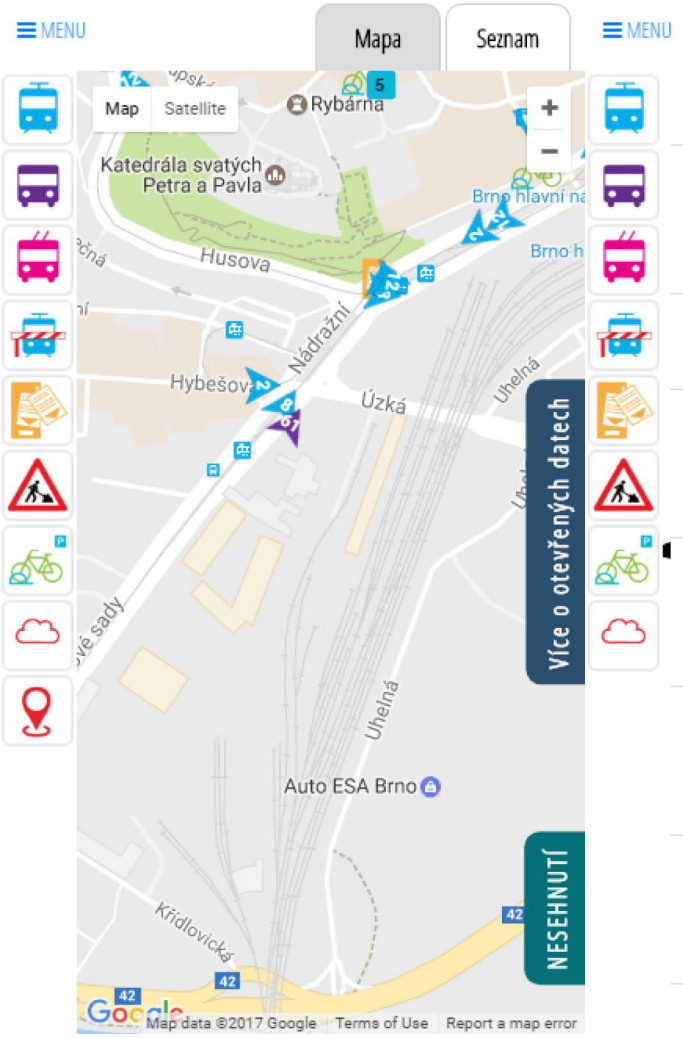
Obr. 1.4 Mobilní aplikace EasyBrno [29]

1.5 Hejbej Brnem

Tato aplikace byla vyvinuta pro nezávislé sociální ekologické hnutí, pracovníky a studenty Ústavu Informatiky Provozně ekonomické fakulty Mendelovy univerzity Brno. „Cílem aplikace Hejbej Brnem je nabídnout občanům a občankám Brna potřebné informace k tomu, aby se snadno pohybovali po Brně – a to různými způsoby – na kole, šalinou, autobusem či pěšky. Dopravní informace a ovzduší ukazují, kde jsou aktuálně největší dopravní problémy a můžou být nápovědou, kterým místům je lepší se vyhnout.“ [30] I přesto, že aplikace je pouze ve formě webové aplikace, je optimalizovaná pro displeje ve velikosti mobilních zařízení. Tato optimalizace je velmi důležitá, bez ní by nebyla uživatelsky přívětivá a mohla by uživatele odrazovat od jejího používání. Přizpůsobenou verzi pro mobilní zařízení lze vidět na obrázku 1.5 a 1.6

- Platforma: Web

- Lokace dat: Brno a okolí
- Datum vzniku: Neznámo
- Url [30]



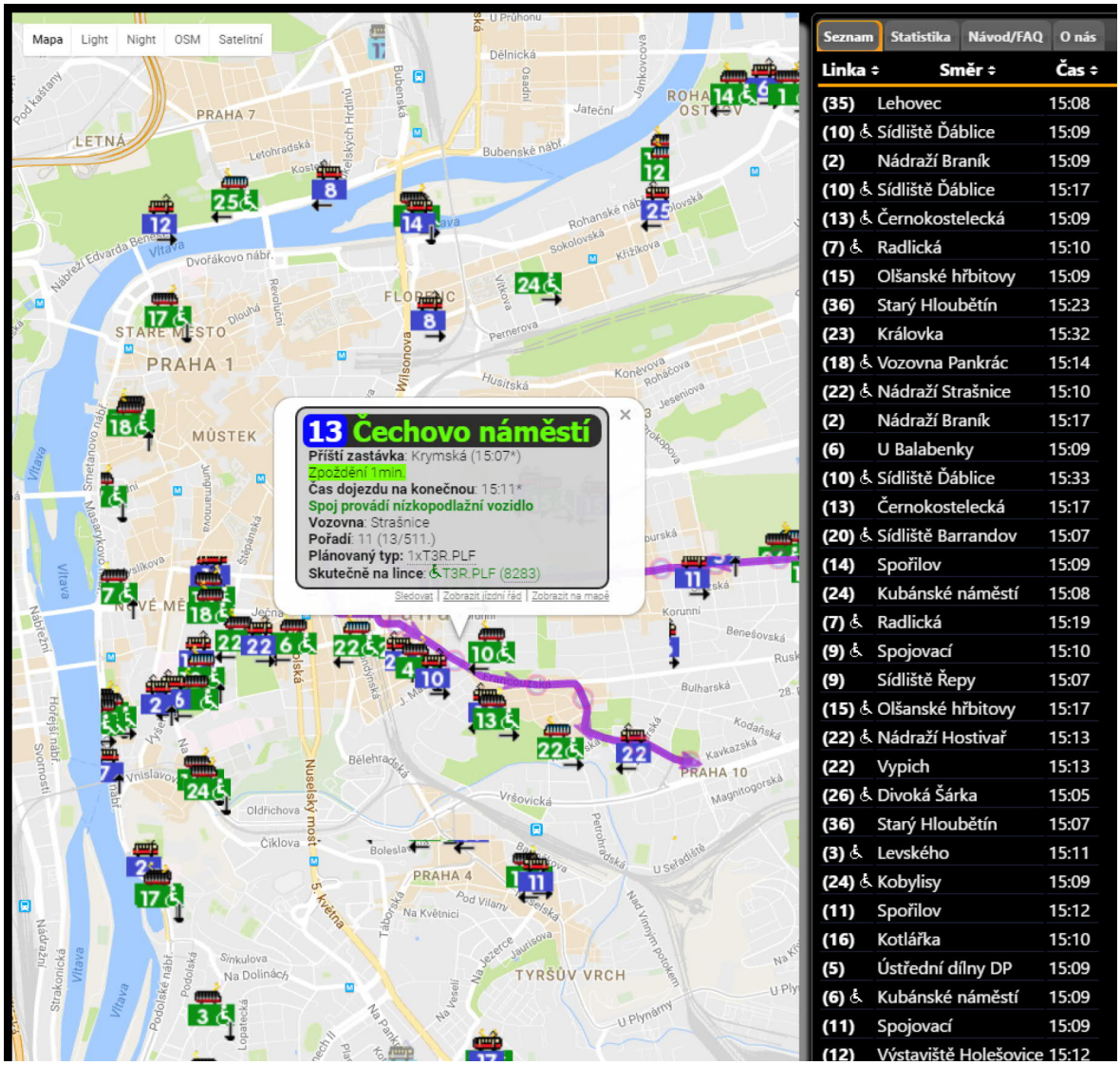
Obr. 1.5 Webová aplikace HejbejBrnem - Mapa [30]

Obr. 1.6 Webová aplikace HejbejBrnem - Seznam [30]

1.6 Mobilní Tabla

„Aplikace slouží ke zobrazení GPS polohy tramvají Dopravního podniku hl. m. Prahy a dodatkových informací o nasazených vozech. Na mapě najdete i v daný čas probíhající výluky provozu.“ [31] Aplikace je k dispozici pouze ve formě webové aplikace a zobrazuje **pouze tramvaje**. Jak můžeme vidět na obrázku 1.7, aplikace není optimalizována pro mobilní zařízeni oproti předchozí aplikaci.

- Platforma: Web
- Lokace dat: Praha (pouze tramvaje)
- Datum vzniku: 2012

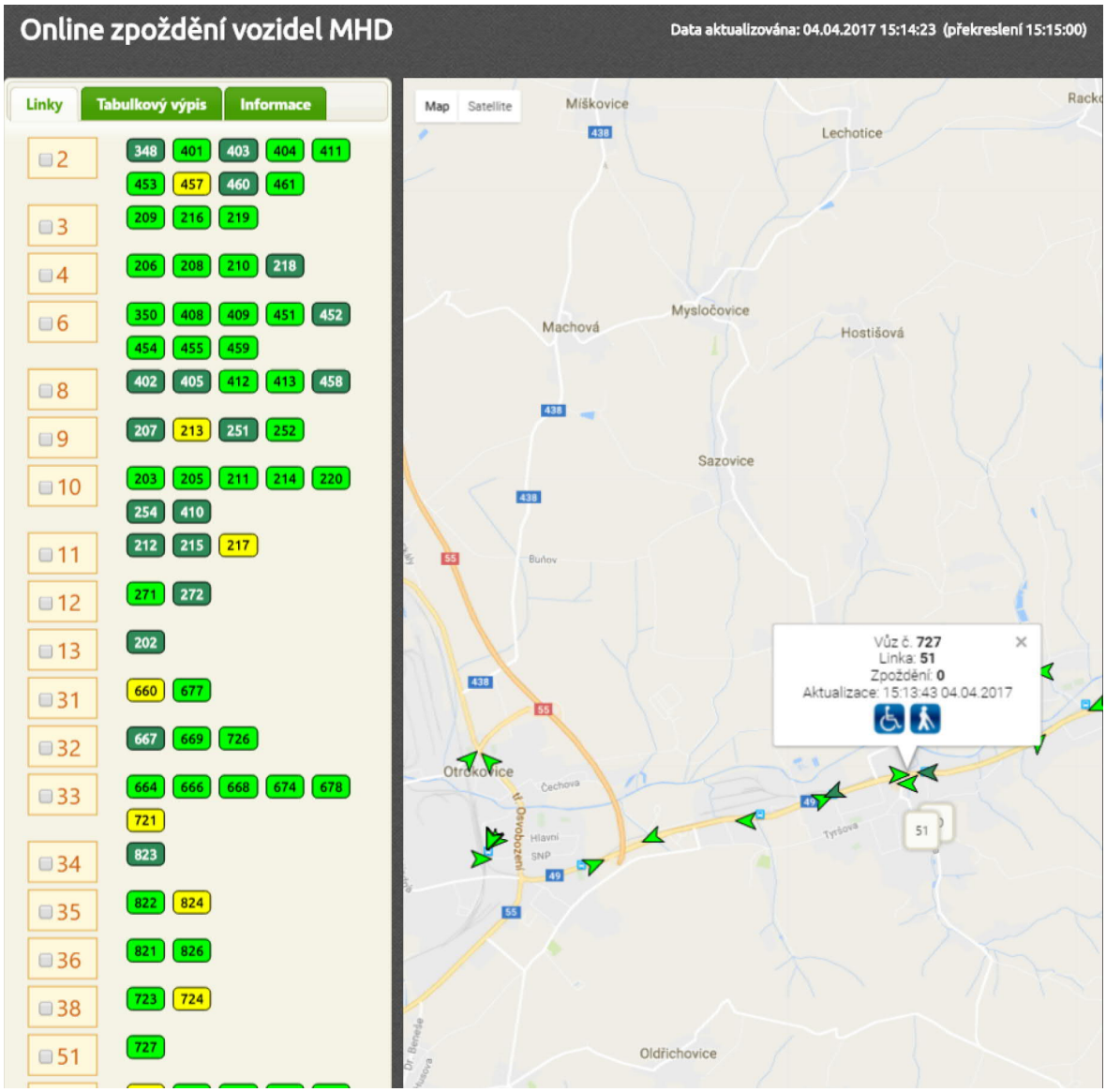


Obr. 1.7 Webová aplikace Mobilní Tabla [31]

1.7 MHD DSZO

Dopravce ve Zlínském kraji nabízí webovou aplikaci na svých stránkách, která disponuje informacemi o aktuálním provozu MHD. Využívá tak svůj informační systém a informuje veřejnost a cestující o aktuálním provozu. [32] „Informace o aktuální pozici vozidel v celé síti MHD zakreslené na mapovém podkladě jsou od konce října – zatím ve zkušebním provozu – k dispozici na adrese www.dszo.cz/online.“ [32] Takto popisuje novinku ředitel Dopravní společnosti Zlín – Otrokovice Josef Kocháň. Na obrázku 1.8 můžeme vidět podobu této aplikace a všimnout si, že stejně jako v předchozím případě není optimalizována pro mobilní zařízení.

- Platforma: Web
- Lokace dat: Zlín
- Datum vzniku: 2015
- Url [32]

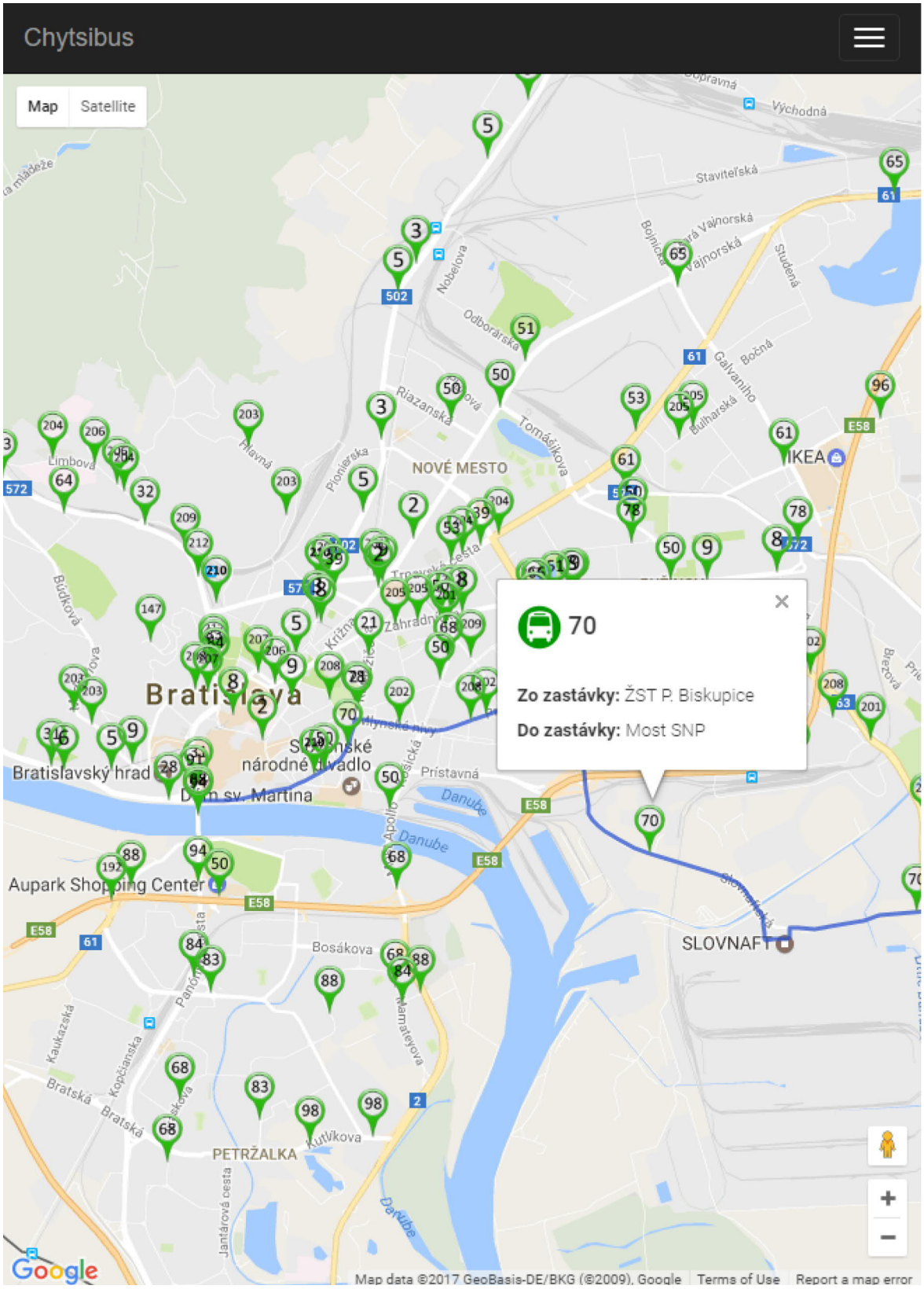


Obr. 1.8 Webová aplikace DSZO [32]

1.8 Chyť si BUS

Tato aplikace je velmi podobná předchozím zmíněným aplikacím. Odlišuje se však jednou velmi zajímavou vlastností. Poloha vozů není získávána z GPS zařízení ve voze, ale je vypočítávána z jízdních řádů, díky čemu lze zobrazovat v téměř v reálném čase jejich polohu, která je však pouze orientační a nemusí souhlasit s reálnou situací. Projekt je poměrně nový a získávání přesných pozic by dodalo na atraktivnosti na tomto projektu.

- Platforma: Web
- Lokace dat: Bratislava (Slovensko)
- Datum vzniku: 2017
- Url [45]



Obr. 1.9 Webová aplikace Chyť si BUS [45]

2 Popis cílových platforem

Pokud bychom chtěli pokrýt co největší část trhu s mobilními telefony, musíme vyvíjet svoje aplikace pro každou platformu zvlášť. Každá platforma je něčím specifická, například použitými technologiemi, strukturou kódu, architekturou, uživatelským rozhraním a programovacím jazykem. Tyto odlišnosti mohou vést ke komplikacím při vývoji.[12]

Další možností je zvolit multiplatformní přístup, tedy programování pro více platforem. Při správném použití nástrojů můžeme dosáhnout až 100% znovupoužitelností kódu mezi platformami. [3]

Abychom mohli začít s vývojem pro všechny základní platformy, je důležité se s nimi seznámit. Pro tuto práci byly zvoleny jako hlavní platformy: Android, iOS a Windows 10 Mobile. Tyto platformy jsou jako hlavní a byly zvoleny podle prodeje zařízení na trhu a jsou znázorněny v tabulce 2.1.

Android tvoří dominantní část trhu s 86,8%, iOS 12,5% a platforma Windows Phone tvoří pouhých 0,3%. [3]

Tab. 2.1 Podíl mobilních platforem [40]

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

2.1 Android

„Nejvíce rozšířeným mobilním operačním systémem je v současnosti Android od společnosti Google.“ [5] Je založen na linuxovém jádře, které poskytuje služby jako je správa paměti a procesů, hardwarové ovladače a jiné. Pro vývoj aplikací je využita technologie Java, která je spouštěna v běhovém prostředí tzv. Dalvik virtual machine. Systém Android je publikován jako opensource. Často se však můžeme setkat s mnoha jeho modifikacemi, které můžou být také opensource v případě různých komunit a nebo v případě větších výrobců jako je Samsung nebo Xiaomi již není opensource. Je to z toho důvodu, že výrobci často přidávají do systému různé svoje aplikace a nádstavby a nechtějí, aby je používal někdo jiný. Aplikace pro Android je možné získávat skrze oficiální obchod Google Play, využít jiný alternativní obchod nebo nahrát aplikaci manuálně do telefonu ve formě instalačního balíčku. „Aplikaci do katalogu může přidat každý, kdo vlastní vývojářskou licenci. Aplikace podstoupí schvalovací proces, který je prováděn automatizovaně a aplikace je většinou dostupná do několika hodin od prvotní

publikace.“ [5]

„V tomto obchodě se dle výzkumu společnosti Statista Inc. nacházelo zhruba 1,5 milionu aplikací. To představuje obrovské množství možností pro koncové uživatele a na druhou stranu to vytváří vysoce konkurenční prostředí pro vývojáře mobilních aplikací.“ [4]

Nevýhodou Androidu, zejména ze strany vývojáře, je jeho velká roztříštěnost verzí. S příchodem nové verze nedochází k automatické aktualizaci na novější verzi. Staré verze tvoří tedy delší dobu většinový podíl a nové verze je obměňují velmi pomalu. Tuto skutečnost potvrzuje fakt, že nejnovější verze Androidu 7, která byla uveřejněna v červenci roku 2016 a teprve na začátku roku 2017 pokořila hranici 1%. Vývojář je tedy nucen při vývoji cílit na nejvíce zastoupenou verzi a často se tak může stát, že nebude moci použít funkce API, které jsou k dispozici pouze v nejnovější verzi. [10]

Tab. 2.2 Podíl verzí Androidu [41]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	10	1.0%
4.1.x	3*Jelly Bean	16	3.7%
4.2.x		17	5.4%
4.3		18	1.5%
4.4	Kitkat	19	20.8%
5.0	2*Lollipop	21	9.4%
5.1		22	23.1%
6.0	Marshmallow	23	31.3%
7.0	2*Nougat	24	2.4%
7.1		25	0.4%

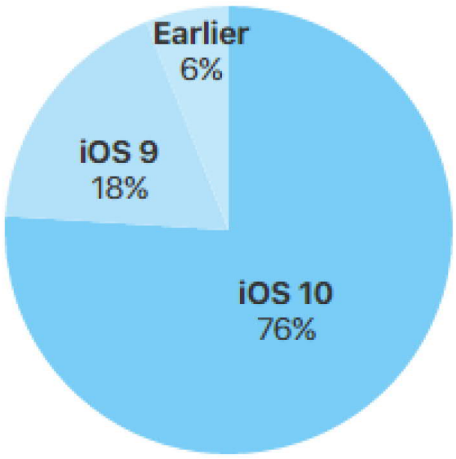
2.2 iOS

„iOS je operační systém firmy Apple, vychází ze systému určeného pro Mac OS a je přizpůsoben pro mobilní zařízení.“ [8] Vyvíjet aplikace pro platformu iOS je možné ve více programovacích jazycích, jsou to Objective-C, což je nadstavba standardního jazyka C rozšířená o možnosti využití objektově orientovaného programování. Dále je to samozřejmě jazyk C, C++ a od roku 2014 swift. Swift je programovací jazyk od společnosti Apple určený pro vývoj na platformách Mac OS X a iOS.

Protože je systém iOS modifikován pro mobilní zařízení, nedisponuje veškerou funkcionalitou jako OS X. Má však navíc dotykovou vrstvu a systém tedy musí být upraven tak, aby jej bylo možné ovládat pomocí dotykového ovládání. [18]

„Tento systém je velmi uzavřený, neumožňuje vývojářům přístup do systému, aplikace je možno instalovat pouze přes oficiální AppStore, kde jsou všechny aplikace přísně kontrolovány, aby neobsahovaly bezpečnostní chyby, jejich kód nebyl škodlivý a neměly nevhodný obsah.“ [8] Za tento striktní přístup je často kritizován, nicméně je to i ku prospěchu věci. App Store není přeplněn duplicitními aplikacemi a navíc nejsou tak často napadeny škodlivým malwarem. [8]

Operační systém iOS na rozdíl od předchozího systému Android má tu výhodu, že po vydání nové verze systému je nabídnuta všem podporovaným zařízením nejnovější verze. Tuto vlastnost může využít vývojář ve svůj prospěch, že se nemusí příliš zaobírat tím, pro jakou verzi iOS bude vyvíjet. Další výhodou může být to, že Apple dodává tento systém jen pro svoje zařízení a není jich tak takové množství jako pro Android.



Obr. 2.1 Zastoupení verzí iOS [42]

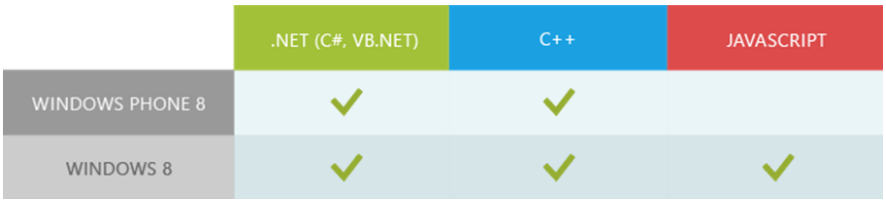
2.3 Windows

2.3.1 Windows Phone

„Windows Phone (často zkracován jako WP) je nástupce dříve populárního systému Windows Mobile (zkracován jako WM), který byl značně pozměněn, a proto došlo i ke změně názvu, ale zůstalo se u číslování verzí.“ [11]

Od uvedení Windows Phone 7 v roce 2010 proběhlo několik aktualizací a číslování majoritní verze se změnilo. S příchodem Windows Phone 8.1 se název změnil zpět na označení Windows Mobile. Pro programování Windows Phone aplikací je možné využít platformu .NET, tedy jazyky C# a Visual Basic .NET. S příchodem novější verze Windows Phone 8 a běhové prostředí Windows Runtime lze využít i C++ a Javascript.

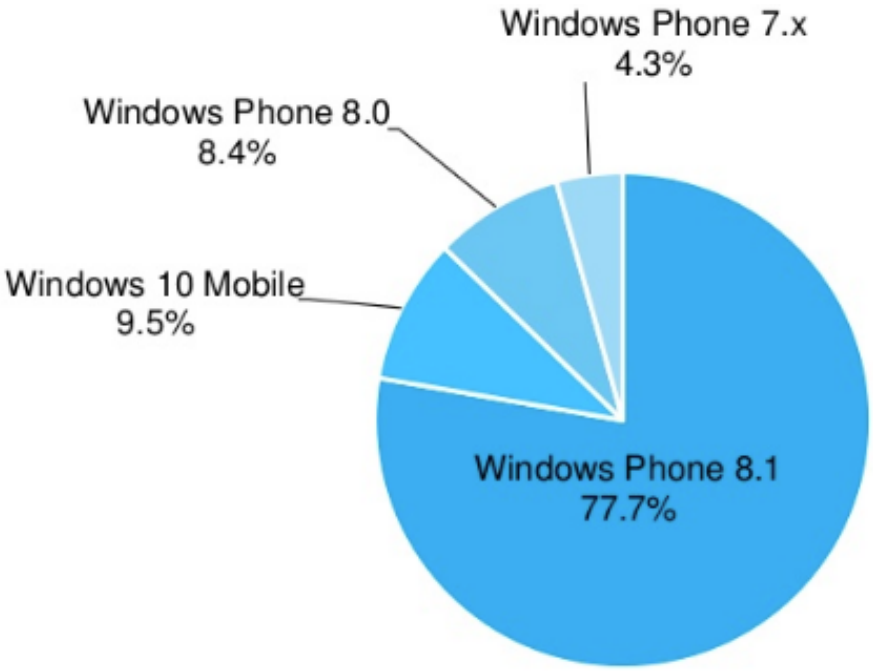
„Windows Phone je momentálně (únor 2016) třetím nejpoužívanějším mobilním operačním systémem“. [6]



Obr. 2.2 Windows Phone programovací jazyky [34]

Systém Windows Phone se velmi liší od předešlých platforem. Jedná se o uzavřený systém, kde aplikace mají přístup pouze do pár vybraných složek. „Všechny ostatní zdroje, jako je databáze pro ukládání aplikačních nastavení, úložiště pro BLOBy nebo dočasné soubory, jsou pro každou aplikaci virtualizované a tudíž pro každou aplikaci jedinečné (takzvaný sandbox)“. [34]

Aplikace vytvořené pro tuto platformu lze instalovat skrze oficiální obchod Microsoft Store. Přesto, že jde o momentálně třetí nejpoužívanější mobilní operační systém, jeho zastoupení je poměrně mizivé oproti Android a iOS. Pravděpodobně je to způsobeno díky malému počtu kvalitních aplikací v obchodě Microsft Store. Zastoupení této platformy na trhu tvoří spolu s Universal Windows Platform pouhých 0,3% a stále klesá. [6]



Obr. 2.3 Zastoupení verzí Windows Phone [43]

2.3.2 Universal Windows Platform

Myšlenka univerzálních aplikací, které je možné spustit na zařízeních Windows s různými rozlišením a vstupními interakcemi, je realizována vytvořením aplikační platformy Universal Windows Platform (UWP). Tato platforma je součástí sjednoceného jádra OS Windows 10 a je dostupná na všech jeho zařízeních. UWP je označována vícero termíny, jako aplikační model, služba a platforma nejen proto, že umožňuje kompilovat a spouštět aplikace. Takto vytvořené aplikace je možné spouštět na mobilech, tabletech, desktopových počítačích a konzoli Xbox One. Návrh této platformy pomáhá docílit toho, že budeme mít jeden aplikační balík spustitelný na zařízeních Windows 10 a distribuovaný ve společném obchodně Microsoft Store.[7]

Obr. 2.4 UWP platformy [44]

UWP vznikla rozšířením aplikačního modelu Windows RT tak, aby aplikace už nebyly omezené jen na určitou vrstvu sady rozhraní API společné pro všechny zařízení, ale mohly přistupovat i k jeho specifickým rozhraním a využít tak všechny jeho schopnosti. Zařízení jsou kategorizované do rodin podle jejich vlastností, využití a hardwarových možností. Pro tyto rodiny zařízení existuje rozšíření v podobě rozhraní API a obsahují charakteristické funkcionality. To znamená, že aplikace využívající jen jádro UWP bude poskytovat stejnou funkcionalitu na každém zařízení, avšak pokud využijeme specifickou vlastnost jako je například posílání SMS zprávy, tak tato funkcionalita bude přístupná jen na zařízeních patřících do rodiny mobilních zařízení. Z tohoto důvodu aplikace už nejsou cílené pro operační systém, ale pro libovolný počet rodin zařízení. [7]

3 Výběr vhodné technologie pro vývoj

3.1 Nativní vývoj aplikací

Nativní aplikace jsou vytvářeny pouze pro konkrétní platformu, a to jak po grafické stránce, tak i svojí strukturou a využitím hardwarových prostředků. Výhoda tohoto přístupu spočívá v tom, že takto naprogramované aplikace jsou obvykle rychlé, spolehlivé a umí efektivně využívat hardware telefonu. „Tyto aplikace jsou napsané v programovacím jazyce dané platformy za použití dostupných oficiálních SDK a dodržují platformou stanovené vývojářské postupy.“ [4]

Pokud zvolíme nativní vývoj aplikací, tak se často postupuje v oddělených týmech podle platforem, kde pro každou platformu se vytváří vlastní implementace uživatelského rozhraní, datového modelu i business logiky aplikace. V podstatě tedy vznikají různé aplikace pro každou platformu. Tyto aplikace sice plní stejnou funkci, nicméně uvnitř mohou fungovat odlišně. To se může případně projevit i na tvorbě serverové části aplikace, kde mohou vznikat různá specifická API pro danou platformu. [4]

Výhody

- Výkon
- Podpora nejnovějších API
- Nativní UI prvky

Nevýhody

- Nemultiplatformnost
- Nutná znalost více programovacích jazyků
- Náročné opravy aplikací

3.2 Hybridní a webové frameworky

Pokud bychom vytvářeli mobilní aplikaci formou webové aplikace, nemůžeme přistupovat k funkcím a službám systému, protože běží v tzv. sandboxu a jsou odděleny od systému. Abychom mohli využívat systémové služby a funkce, je nutné obalit tuto aplikaci do tzv. „wrapper aplikace“, která obsahuje pouze instanci webového prohlížeče.

„Každá platforma dnes obsahuje něco, čemu se odborně říká „WebView“, nebo-li instanci prohlížeče. [14] Tedy můžeme říct, že se jedná o obrazovku prohlížeče, který načítá webový obsah, ať už offline či online. Zároveň wrapper aplikace může interagovat s uživatelem na různé akce, případně zasílat notifikace atd. [14]

Další indispozicí webových aplikací je, že je nelze nahrávat do oficiálních obchodů a uživatel tak často může zapomenout o existenci aplikace. [14]

Výhody

- Multiplatformnost
- Snadné testování
- Webové technologie

Nevýhody

- Vyšší požadavky na hardware
- Možná nedostupnost API
- Chybí nativní UI prvky

3.2.1 PhoneGap

Tento framework využívá webových technologií jako je HTML, CSS a Javascript a umožnuje tak vývoj multiplatformních aplikací. PhoneGap vychází z open-source projektu Apache Cordova a podporuje přední platformy Android, iOS a Windows Phone. Principem tohoto frameworku je zobrazování webových stránek, které se pak zobrazují pomocí tzv. WebView. Tyto stránky jsou zabaleny do nativní aplikace, která může přistupovat ke službám a prostředkům. V případě, že je potřeba přistupovat k hardwarovým prostředkům jako je GPS, kamera, adresář kontaktů atd. využívá se pluginů z Cordovy. [14] [4]

3.2.2 Appcelerator Titanium

Jedná se o open-source framework, který slouží k vytváření multiplatformních mobilních aplikací pomocí HTML, Javascriptu a Titanium SDK. Aplikace je pak přeložena do nativního kódu pro mobilní platformy. [8]

Aplikace napsaná pomocí Titanium frameworku spouští nativní aplikace propojené s výkonným javascriptovým prostředím. „V JavaScriptovém prostředí běží vytvořená multiplatformní aplikace, ovšem její uživatelské prostředí se renderuje v nativní aplikaci.“ [4]

3.3 Mobilní web

Mobilní web je vhodný tehdy, pokud nevyžadujeme žádnou speciální funkcionalitu, pouze uživatele připojeného k internetu a přestylování webové aplikace pro mobilní

aplikace. „Tento přístup se používá hlavně u velkých webových aplikací, které není možné správně optimalizovat pro mobily.“ [14]

Pokud by však aplikace kopírovala funkcionalitu z již hotové webové aplikace, je vhodné zauvažovat, zda není výhodnější optimalizovat webovou aplikaci pro mobilní zařízení, případně zvolit jiný přístup. [14]

Výhody

- Rychlost vývoje
- Univerzálnost
- Není nutné instalovat aplikaci do telefonu. (může být i nevýhodou)
- Aplikace není potřeba distribuovat přes obchod třetích stran

Nevýhody

- Mizerný „uživatelský prožitek“
- Chybí napojení na API telefonu
- Chybí nativní prvky UI

3.4 Xamarin

Xamarin je nástroj pro multiplatformní vývoj aplikací. Používá jazyk C#, který následně pracuje na platformách Android, iOS, Windows Phone a Universal Windows Platform, kde je jazyk C# nativně podporovaný. Pomocí tohoto nástroje je možné vytvořit aplikace, které jsou dostatečně rychlé a výkonné, jako například hry. [13]

„Xamarin je trochu odlišný od dříve představených nástrojů, protože každá platforma má vlastní implementaci. Produkt Xamarinu je tak rozdělen na 3 hlavní podprodukty Xamarin.Android, Xamarin.iOS a Xamarin.Forms.“ [4]

Nativní API pro Android i iOS bylo přeloženo tak, aby jej bylo možné volat i z jazyka C# díky API bindingu. Jak Xamarin.Android tak i Xamarin.iOS jsou oddělené projekty, takže je možné psát zvlášť jak nativní uživatelské rozhraní, tak i aplikační logiku, kde může být požadováno chování funkcionality na jedné platformě odlišné než na druhé. Zároveň je možné psát i společnou logiku aplikace a tu využít i v případě Windows Phone a obou předchozích platforem. [4]

3.4.1 Xamarin.Android

Projektová část Xamarin Android používá ke svému běhu open-source projekt Mono. „Mono je napsané v programovacím jazyce C a běží současně vedle vlastního běhového prostředí Androidu nad linuxovým jádrem. [4] Projekt Mono for Android je neustále vyvíjen a musí současně reflektovat všechny změny, které byly provedeny v původní Android Java knihovně v každé nově vydané verzi OS Android. Díky existenci Mono for Android projektu je Xamarin Android schopen poskytovat prostředí, které je velmi podobné jako při nativním vývoji aplikací pro Android.

Při vývoji pro Android lze zvolit dvojího přístupu k systémovým prostředkům. První možností je použít standardní knihovny .NET. Další možností je využití API poskytovaného běhovým prostředím Androidu, které je k dispozici i v jazyce C# pomocí tzv. API bindingu [4]

3.4.2 Xamarin.iOS

„Xamarin.iOS je Framework, který nám umožňuje vytvářet aplikace pro operační systém iOS“ [6] Aby bylo možné provést sestavení, je nutné vyvíjet na počítači se systémem Mac OS nebo alespoň mít k dispozici stroj s tímto systém a vývojářskými nástroji xCode a iOS SDK. Pomocí nástroje Build Host Agent lze provést sestavení i vzdáleně. Xamarin.iOS na rozdíl od Xamarion.Android projektu funguje naprosto jinak. Nástroj Mono není spuštěn vedle běhového prostředí, jako v případě Android projektu, ale je využita funkce Ahead-of-Time compiler již během kompilace kódu. Tato funkce má za úkol přeložit kód do asemblerového kódu pro procesory postavené na architektuře ARM.

3.4.3 Xamarin.Forms

Jedná se o multiplatformní framework, slouží především k vytváření aplikací, které nepotřebují specifickou funkcionalitu pro každou platformu. Umožňuje napsat jednu funkcionalitu a tu použít na všech platformách. Zejména formulářové aplikace jsou díky tomuto frameworku skvěle přizpůsobeny a na každé platformě vypadají tak jako jim říkají pravidla pro návrhu UI dané platformy. Xamarin Forms nabízí spoustu již hotových komponent, díky čemuž je vývoj takovýchto aplikací velmi rychlý. Nicméně komponenty mají omezenou funkcionalitu a pokud požadujeme jinou funkcionalitu, jsme schopni si napsat vlastní funkce nebo i celou komponentu. [15]

„Xamarin.Forms je framework, který ještě dále integruje Xamarin.Android, Xamarin.iOS a vývoj pro Windows Phone a kromě datového modelu a business vrstvy unifikuje a umožňuje sdílet i aplikační vrstvu, která se stará o otevírání nových obrazovek, navigaci mezi obrazovkami a předávání dat.“ [4]

Obr. 3.1 Xamarin Forms [4]

3.5 Výběr technologie

3.5.1 Nativní aplikace

Aplikace vyvíjené nativně dodávají mnohem lepší uživatelský zážitek než jiná řešení, navíc jsou vyvíjeny v navrženém jazyce a obvykle jsou výkonnější. Pokud bychom zanedbali cenu vývoje, je tento způsob většinou nejvhodnějším způsobem. [4]

3.5.2 Použití webových technologií

Pokud budeme uvažovat o vývoji pomocí webových technologií, je vhodné je využít u menších aplikací, které nejsou příliš provázané s hardwarovými prostředky a funkcemi telefonu. Nejčastější využití najdeme v aplikacích typu RSS čtečky, novinové aplikace, propagační aplikace. [4]

3.5.3 JavaScriptové aplikace

Aplikace vytvořené pomocí javascriptových frameworku využívají nativní uživatelské rozhraní a jsou často závislé na datovém přenosu, který využívají ke komunikaci s API nějakého serveru. Takto vytvořené aplikace jsou vhodné v případě, že zobrazujeme nějaká data poskytované z API serveru, například kurzovní lístek.

3.5.4 Xamarin

Xamarin je výkonově velmi blízký nativnímu vývoji. Implementace pro iOS a Android je velmi rozdílná a lze tedy plně oddělit vývoj uživatelského rozhraní pro každou platformu. Tento typ vývoje se hodí pro téměř jakoukoliv aplikaci, navíc lze vyvíjet levně pro všechny přední platformy. [4]

Tento způsob se zdá jako nejvhodnější způsob vývoje mobilní aplikace pro tuto práci. Vzhledem k tomu, že potřebujeme vytvořit aplikaci, která bude zpracovávat data velmi rychle, je vhodný způsob nativní vývoj. Ten by však neodpovídal zadání této práce. Hybridními frameworky bychom dosáhli multiplatformního vývoje, nicméně pro slabší telefony by to mohl být problém z pohledu požadavků na výkon.

4 Použité technologie

4.1 WSDL

„WSDL je jazyk založený na XML. Popisuje konkrétní webovou službu, její umístění, metody, parametry a další informace nutné pro přístup a komunikaci.“ [36] Tak jako XML má i WSDL přesně danou strukturu a pro programy je dobře zpracovatelný. WSDL soubor je jakési rozhraní pro SOAP klienta, pomocí kterého vygenerujeme kód v daném programovacím jazyce. Po zpracování WSDL souboru jsme schopni SOAP klientem volat nabízené funkce vzdálené služby. [21] [36]

4.2 .NET Core

Díky této nové technologii neznamená vývoj na platformě .NET pouze Windows. NET Core je víceúčelová vývojová platforma vedená společností Microsoft. Jejím velkým přínosem je multiplatformnost, modulárnost a je k dispozici jako open-source. Současně lze tento framework provozovat na operačních systémech Windows, macOS a Linux, ale může být naportován i pro jiné operační systémy. „Skládá se z běhového prostředí CoreCLR a modulární sady assemblies. Nový framework .NET Core je nyní dostupný v několika implementacích, přičemž webové vývojáře bude zajímat především ASP.NET Core, což je unifikace stávajícího ASP.NET MVC + ASP.NET Web API.“ [16]

„Nový ASP.NET Core nabízí pohodlnější vývoj, lepší podporu různých balíčkovacích systémů (bower, gulp, grunt) a vyšší výkonnost.“ [16]

„Z důvodu multiplatformní podpory lze komunikovat s CoreCLR skrze nástroje .NET Core CLI (Command Line Interface) Tools. CLI Tools je možné používat pomocí příkazové řádky, např. skrze příkazy dotnet new, dotnet restore atd.“ [16]

Hostování .NET webových aplikací doposud bylo poměrně nákladné ve srovnání s jinými jazyky, které je možné hostovat na OS Linux. Stačí nám k tomu nainstalovaný Apache nebo nginx, který slouží jako proxy a předává data z/do naší aplikace. [35]

Musíme také zdůraznit, že ASP.NET Core existuje zároveň s klasickým ASP.NET a můžeme předpokládat, že tento nový framework bude postupně vytlačovat stávající .NET. [16]

4.3 Xamarin GoogleMaps

Tato knihovna poskytuje funkce, které jsou nativně k dispozici pro iOS a Android. Systém iOS využívá ve výchozím stavu MapKit, tedy Apple maps. Ty však byly zaměněny za Google maps, kvůli jednodušší implementaci a společnému API s Androidem.

V případě Windows Phone a UWP, kde není nativní podpora Google maps, jsou poskytovány Bing maps. Ty však působí staromódním dojmem a některé oblasti, které jsou v Google maps podporovány, zde nejsou vůbec. Celá knihovna je založena na Xamarin.Forms.Maps, která však disponuje minimem funkcí. Tato knihovna je tedy jakýmsi rozšířením o chybějící funkce. Celý projekt je opensource a právě díky tomu jsme schopni doimplementovat funkce, které v ní stále chybí a jsou potřebné pro tuto práci. Nevýhodou opensource projektu může být však to, že jej nemáme kompletně pod svou správou a jsme tak závislí na autorovi, kdy uvolní další aktualizaci balíčku. Při větší uživatelské základně to nemusí trvat příliš dlouho, ovšem při menším počtu uživatelů to může být i velmi dlouho.

II. ANALYTICKÁ ČÁST

5 Zdroje dat

5.1 Získávání dat

Abychom byli schopni zobrazovat data o spojích, musíme nejdříve si opatřit vhodný zdroj. Pravděpodobně vhodným zdrojem bude řídící středisko daného dopravce. Data z řídícího střediska však nejsou zpravidla veřejně dostupná a je na daném vývojáři a dopravci, jak se dohodnou na zpřístupnění a poskytování dat. Na základě zkušenosti se získáváním dat v této práci můžeme tvrdit to, že čím menší firmu tvoří dopravce, tím lehčí je se dohodnout na zpřístupnění dat. Nejčastější způsob předávání dat je pomocí WSDL nebo generovaný výstup ve formátu CSV, JSON. [9]

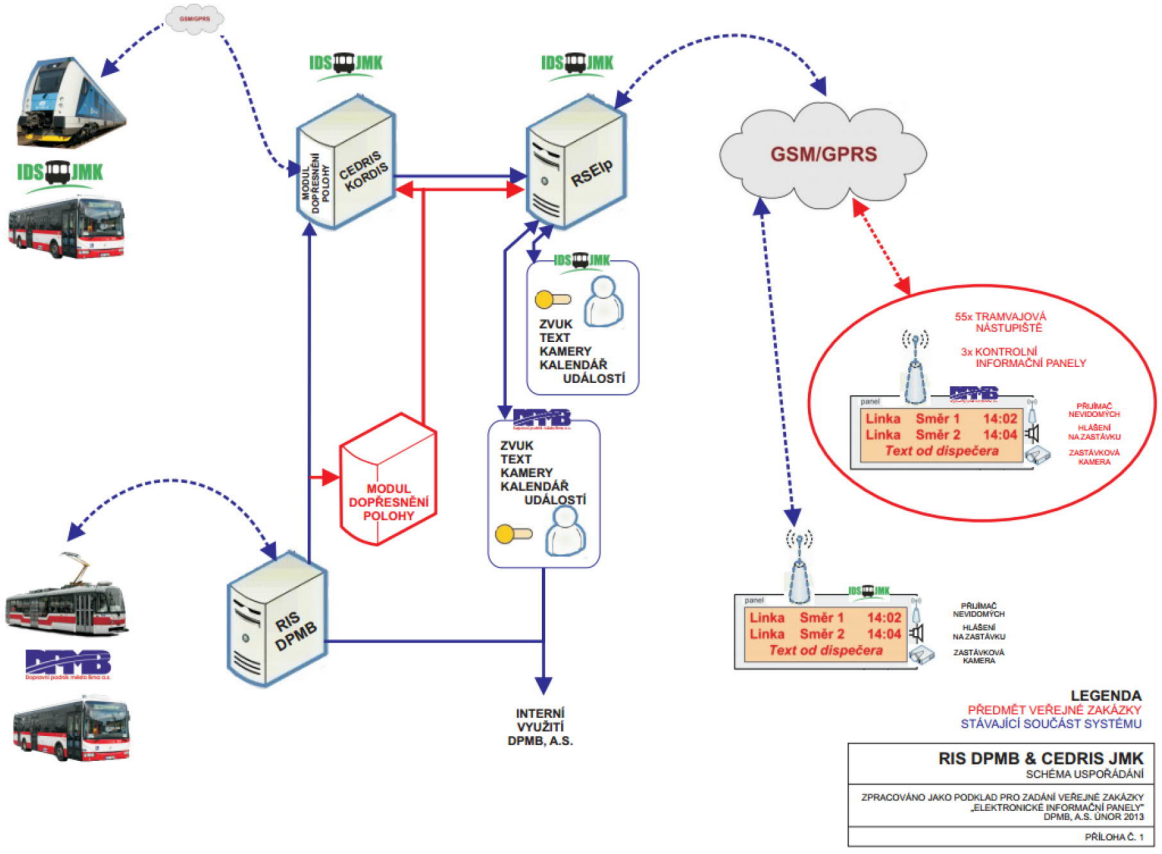
5.2 Kordis

„KORDIS JMK byl založen za účelem vykonávání dvou hlavních činností: koordinace základní dopravní obslužnosti na území Jihomoravského kraje a přípravy, realizace a provozování integrovaného dopravního systému postupně na celém území Jihomoravského kraje.“ [19]

Díky řídícímu středisku má nástroje, které mu umožňují sledovat a vyhodnocovat vývoje přepravních potřeb, navrhovat efektivně jízdní řády a optimalizovat vedení linek. Zavedení řídícího a informačního střediska došlo od roku 2004, kdy byl oficiálně nasazen ke zlepšení, jak pro obyvatele využívající městskou hromadnou dopravu, tak i zaměstnance využívající tento systém. Kordis má tedy plnou kontrolu nad organizací provozu MHD, sledování stavu provozu dispečerem v reálném čase, řízení odjezdů, výhybek, hlášení datovou komunikaci s vozidly.

Při vzniku centrálního dispečinku si vzal za vzor již existující dispečink DPMB - RIS. Ten ve své době byl jedinečný v celém středoevropském regionu. Řídící informační systém DPMB získává ze všech vozidel data a využívá je dále pro řízení dopravy.

CED přebírá data z toho systému a disponuje tak informacemi o poloze, aktuální odchylce od jízdního řádu u všech vozidel veřejné dopravy v IDS JMK. [19] [20] Informační systém CED nejen že integruje další dopravce, ale zároveň doplňuje některé chybějící funkcionality. Například "modul pro dopřesnění polohy"zajišťuje zpracování dat ze systému RIS, zjistí aktuální polohu vozidla a jeho jízdní řád. Z těchto informací je schopen dopočítat odchylku o zpoždění.



Obr. 5.1 Schema RIS a CEDRIS [38]

5.2.1 Smluvní kontrakt

V případě získávání dat od subjektu Kordis JMK v této práci bylo nutné sjednat dohodu o spolupráci. Na základě této smlouvy je autor oprávněn využívat data o polohách zastávek IDS JMK, vozidel provozovaných na linkách IDS JMK, označníků a jejich názvů a časech aktuálních odjezdů.

Aby došlo ke zpřístupnění dat, v případě poskytovatele dat Kordis bylo nutné předat technickému oddělení IP adresu, ze které budou přicházet požadavky na získávání dat. Je to z důvodu bezpečnosti, aby společnost měla přehled, kdo má přístup k jejich datům. Následovalo předání technických parametrů, jako je způsob komunikace se serverem poskytující tato data. Komunikace se serverem v tomto případě probíhá pomocí tzv. "Web Service Description Language"(WSDL) Pomocí nástrojů, jako je Visual Studio, můžeme vygenerovat třídu, která popisuje dané objekty a umožní nám komunikovat s danou službou.

Z vygenerovaných tříd můžeme použít klienta, pomocí kterého získáme požadovaná data. Například pokud chceme získat data o vozidlech a jejich polohách, použijeme následující kód:

```
var trafficStateClient = new TrafficStateClient();
var traffic = await trafficStateClient.GetTrafficStateAsync();
```

Nyní tedy máme potřebná data, takže si popíšeme, co obsahují a jakou mají strukturu.

5.3 Struktura dat

Systém poskytuje velmi mnoho informací a můžou být využity při rozšiřování této práce, nás však momentálně bude zajímat následující objekt, který popisuje jednotlivá vozidla.

5.3.1 Vozidla

Abychom se získanými daty mohli dále pracovat, je potřeba pochopit, co vlastně obsahují. Uvedeme si tedy objekt, který reprezentuje vozidlo vygenerovaného z poskytované webové služby v jazyce C#

```
public partial class Entry : object
{
    private ushort CarNumField;
    private short DelayInMinsField;
    private ushort FinalStopIDField;
    private bool IsBarrierLessField;
    private byte LastPostIDField;
    private ushort LastStopIDField;
    private float LatitudeField;
    private uint LineIDField;
    private float LongitudeField;
    private ushort RouteIDField;
    private byte StateField;
    [System.Runtime.Serialization.DataMemberAttribute()]
    public ushort CarNum
    {
        get
        {
            return this.CarNumField;
        }
        set
        {
            this.CarNumField = value;
        }
    }
}
```

Číslo vozidla reprezentuje unikátní číslo každého vozidla, podle něj dispečer získá všechny potřebné informace. Zpoždění je atribut, který je dopočítáván podle jízdního řádu a unikátního čísla vozidla. ID stanic je ID, pomocí kterého jsme schopni dohledat další informace o stanici, jako je její poloha nebo název. Pomocí atributu zeměpisné šířky a délky jsme schopni určit přesné místo na zemském povrchu. Název linky je obyčejně nějaké číslo spoje. Název spoje však často můžeme pozorovat i s písmeny např. N99. Pro takové případy je nutné mít aliasy linky, které nahrazují číslo za nějaký řetězec. Pomocí ID trasy jsme schopni dohledat body, kterými spoj projíždí. Jedná se o objekt, který nese informace o zeměpisné šířce a délce.

Vysvětlivky k jednotlivým datům

- CarNum = Číslo vozidla
- DelayInMins = Zpoždění v minutách
- FinalStopID = ID Cílové stanice
- IsBarrierLess = Bezbarierové vozidlo
- LastStopID = Poslední navštívená stanice
- Latitude = Zeměpisná šířka
- Longitude = Zeměpisná délka
- LineID = Název linky
- RouteID = ID trasy
- State = Stav vozidla

5.3.2 Zastávky

Oproti vozidlům data o zastávkách jsou získávána jiným způsobem. Protože vozidla jsou aktualizována ve vteřinovém intervalu a jsou to tedy dynamická data, je tedy logické je získávat pomocí webových služeb. Naproti tomu zastávky, které se téměř nemění, jsou data statická a není tedy potřeba je aktualizovat tak často. Data nesoucí informace o zastávkách jsou ukládána na FTP serveru ve formě textového souboru. Ten je však součástí zip souboru, takže je nutné jej nejprve rozbalit. Pokud bychom analyzovali tato data, bude nás zajímat informace nesoucí ID zastávky a také její název, případně zóna, do které spadá. Jeden řádek z tohoto souboru vypadá následovně.

```
01001 101 'Achtelky' 'Achtelky' 'Achtelky' 'Achtelky' 'Achtelky'
```

Vysvětlivky k jednotlivým datům

- ID zastavky
- číslo zóny
- 5x název zastávky (používá se první název)

5.4 Dopočítávání chybějících dat

Jak si můžeme všimnout, ve zmiňovaných datech je téměř vše, co bychom mohli potřebovat pro zobrazování spoje na mapě. Chybí nám však jedna zásadní informace a to je, jakým směrem zrovna spoj jede. Tuto informaci systém Kordis neposkytuje a to i přesto, že RIS DPMB touto informací disponuje. Je tedy nutné ji dopočítat. Naštěstí k aktualizaci dat přímo ze spojů dochází velmi často a vypočítaný azimut je tedy poměrně přesný a pro naše účely je dostačující.

5.4.1 Azimut

Azimut spočítáme podle následujícího vzorce: $\theta = \text{atan2}(\sin\Delta\ \lambda \cdot \cos\ \varphi_2, \sin\ \varphi_1 \cdot \cos\ \varphi_2 - \sin\ \varphi_1 \cdot \cos\ \varphi_2 \cdot \cos\ \Delta\ \lambda\ )$ kde φ_1, $\lambda 1$ je počáteční bod φ_2, $\lambda 2$ je konečný bod a $\Delta\ \lambda$ je rozdíl v zeměpisné délce.

V programovacích jazycích je funkce atan2 definována tak, že vrací hodnoty v intervalu $\langle\ -\ \pi\ ;\pi\ \rangle$. Proto je nutné výstup z této funkce normalizovat. To provedeme tak, že nejprve převedeme radiány na stupně a poté k θ přičteme 360 a použijeme dělení se zbytkem. Vzorec po normalizaci tedy bude vypadat následovně $\theta + 360\ \%\ 360$. [22]

5.5 DSZO

Zpřístupnění dat v případě dopravní společnosti ve Zlíně nebylo natolik složité jako v předchozím případě. Na webu společnosti můžeme najít odkaz na webovou aplikaci, která zobrazuje reálný provoz a zpoždění vozidel MHD. Data na webu však nejsou dostatečná pro naše použití. Z tohoto důvodu bylo důležité navázat spolupráci a požádat tak o surová data k dalšímu zpracování. Data jsou poskytována jako výstup skriptů ve formě CSV oddělené středníkem. Tato data jsou generována pomocí PHP skriptu na straně serveru.

5.5.1 Vozidla

Abychom i zde mohli porozumět poskytovaným datům, je důležité nejprve pochopit, co vlastně reprezentují.

Příklad jednoho řádku z CSV:

```
201;4802;4;10;10;6;1904;12302;2602;17.602899999999998;49.20973;
16:18:42 10.03.2017;67
```

- Unikátní číslo vozu
- ID poslední navštívené zastávky
- Číslo linky vozu
- Číslo služby
- Číslo kurzu služby
- Číslo řidiče
- ID počáteční stanice
- ID konečná stanice
- Zeměpisná šířka
- Zeměpisná délka
- Datum a čas odeslání paketu z vozu
- Azimut vozu

Jak si můžeme všimnout, na rozdíl od dat ze systému poskytovaným Kordis, máme k dispozici azimut. Tato informace je oproti předchozímu systému poměrně zásadní, protože data nejsou aktualizována tak často a dopočítávání by bylo velmi zkreslené.

5.5.2 Zastávky

Získávání dat o zastávkách je prováděno podobně jako v případě vozidel. I přesto, že jsou data dynamicky generována, můžeme je považovat jako statická data, protože se téměř nemění. Nyní si popíšeme, co data o zastávkách obsahují z následujícího záznamu.

```
022;01;Janušticе,tenis.kurt;49.2393608;17.6890202;1;0;0;0;AB
```

Podobně jako u Kordisu máme zde ID zastávky, které nám označuje unikátní číslo, nicméně přibyl nám označník číslo sloupku, který může říkat to, že se jedná třeba o zastávku se stejným názvem, ale jinou polohou nebo se může jmenovat i úplně jinak. Kromě zmiňovaného názvu zastávky nám přibyly informace jako je zeměpisná šířka a délka daného označníku a atributy udávající příznak zastávky na znamení, případně i pásmo, což je synonymum k zóně jako je v předchozím případě.

- ID zastávky
- Číslo sloupku
- Název zastávky
- 49.2xxx – GPS délka označníku
- 17.6xxx – GPS šířka označníku
- příznak celodenní zastávky na znamení
- příznak zastávky na znamení pouze v čase 21 - 4 hod.
- příznak celodenního nástupu pouze předními dveřmi
- příznak bezbarierové zastávky
- seznam všech pásem, na kterých zastávka může být

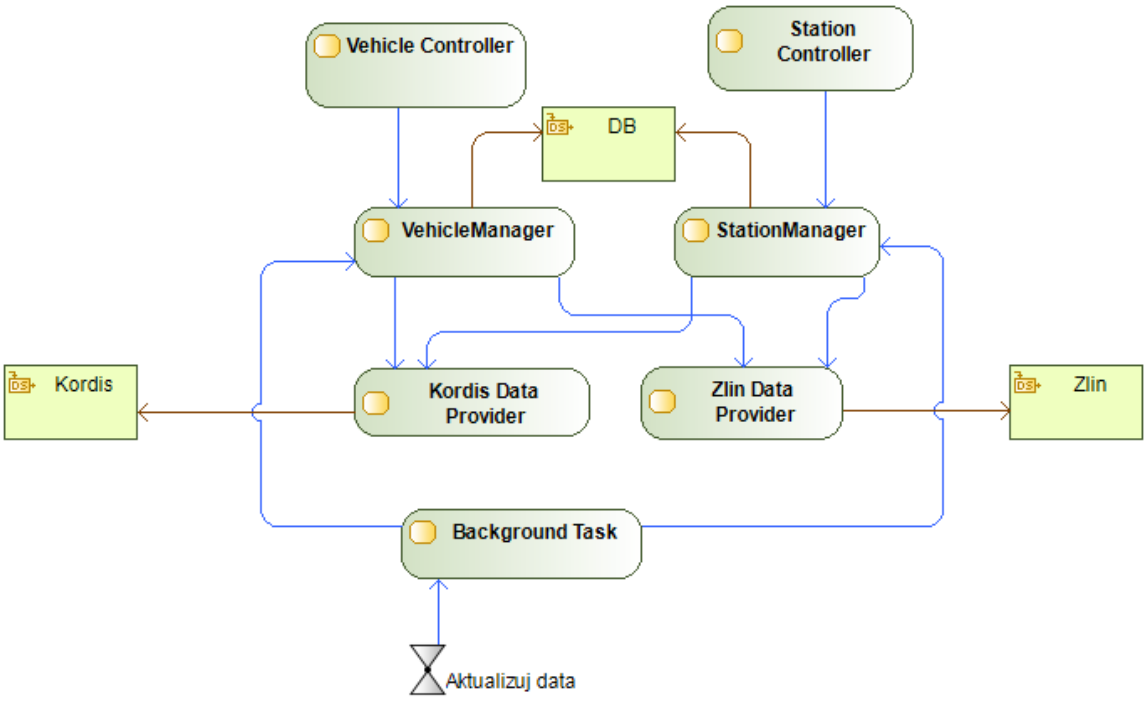
III. PROJEKTOVÁ ČÁST

6 Serverová aplikace

Serverová aplikace má jeden hlavní úkol a tím je poskytování dat pro mobilní aplikaci. Při vývoji bychom měli myslet na to, že konzumentem našich dat je právě mobilní zařízení. Z toho důvodu bychom měli poskytovat pouze data, o které si mobilní aplikace požádá. Tedy odpověď od serveru by měla být poskytována v co možná nejkratším čase a zároveň s co nejmenší možnou velikostí.

Přesto, že naše zadání je vytvořit multiplatformní mobilní aplikaci, byl zvolen podobný přístup i k vývoji serverové aplikace. K tomuto rozhodnutí došlo zejména proto, že provoz linuxových serverů je levnější a mnohdy jsou i výkonově více optimalizované. Zároveň lze využívat platformu Windows pro hostování aplikace, která může být v některých případech výhodnější. Toto kritérium splňuje zmiňovaná technologie .NET Core popsána v předchozí kapitole.

6.1 Diagram



Obr. 6.1 Activity diagram server API

6.2 Implementace serverové aplikace

Serverová část je velmi důležitou částí celého projektu. Bez něj by mobilní část nemohla fungovat. Jádro serverové využívá návrhový vzor Model – View – Controller projekt

založený na ASP.NET Core Web Application šabloně. Hlavní funkční část tedy zastřešuje controller, který odpovídá na dotazy klienta a vrací model, který představuje požadovaná data. View zde je využito pouze pro úvodní stránku, pro kterou však momentálně není vytvořen obsah. Abychom data správně oddělili, je potřeba vytvořit 2 controllery. Jeden z nich se stará o vozidla a informace o nich a další o zastávky.

6.2.1 Nastavení

Abychom mohli jednoduše aplikaci přesunout na jakýkoliv server a nebylo zapotřebí zasahovat do zdrojového kódu aplikace, byl vytvořen konfigurační soubor ve formátu JSON s názvem appsettings.json

```
"ClientConfiguration": {
    "ZlinBaseAddress": "http://server.cz",
    "HttpTimeout": 5,
    "UseSslCertificate": false,
    "KordisFtpAddress": "ftp.server.cz",
    "KordisFtpLogin": "login",
    "KordisFtpPassword": "password"
  },
"ConnectionStrings": {
    "Mysql":
        "server=localhost;userid=user;pwd=password;port=3306;database=db;sslmode=none;"
}
```

Konfigurační soubor obsahuje dvě sekce nastavení. První sekce obsahuje URL adresu serveru, odkud se mají stahovat data. V případě dopravního podniku DSZO se navíc použije i timeout http klienta a příznak použití certifikátu.

Další řádky nastavení platí v případě, že se použije poskytovatel dat Kordis. Jsou to autentizační údaje pro připojení k FTP serveru, kde jsou uložena statická data, zejména názvy zastávek a jejich ID.

Druhá sekce poskytuje tzv. connection string do databáze. Tedy název serveru, kde je databázový server umístěn, jméno uživatele s oprávněním do databáze, heslo a název databáze.

Konfigurační soubor je načten pouze jednou při startu aplikace pomocí následujícího kódu

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<ClientConfiguration>(
        Configuration.GetSection("ClientConfiguration"));
    services.AddDbContext<MhdDbContext>(
        options => options.UseMySql(Configuration.GetConnectionString("Mysql"))
    );
}
```

Sekce z konfiguračního souboru je mapována na námi vytvořenou třídu typu ClientConfiguration. Data v konfigurační třídě pak můžeme použít následovně v dalších službách a třídách díky Inversion of Control. Do konstruktoru naší třídy vložíme typ naší třídy a

.NET framework díky reflexi a předem registrované třídy sám dosadí vhodnou instanci třídy.

```
public class ZlinClient
{
    private static HttpClient _httpClient;
    private ClientConfiguration _clientConfiguration;
    public ZlinClient(IOptions<ClientConfiguration> clientConfiguration)
    {
        _clientConfiguration = clientConfiguration.Value;
    }
    private HttpClient GetHttpClient()
    {
        if (_httpClient == null)
        {
            _httpClient = new HttpClient()
            {
                BaseAddress = new Uri(
                                _clientConfiguration.ZlinBaseAddress),
                Timeout = TimeSpan.FromSeconds(
                    _clientConfiguration.HttpTimeout)
            };
        }
        return _httpClient;
    }
    public async Task<string> FetchUrl(string relativeUrl)
    {
        var response = await GetHttpClient()
            .GetStringAsync(
                new Uri(relativeUrl, UriKind.Relative));
        return response;
    }
}
```

Tato třída zajišťuje to, že budeme mít k dispozici pouze jednu instanci HTTP klienta a nebudou se tak zbytečně otevírat a zavírat spojení na server, ale budeme mít vždy jen jedno spojení. Z tohoto důvodu existuje instance této třídy jen jedenkrát jako singleton. Na stejném principu funguje i třída KordisFtpClient, která je využívána pro stahování dat z ftp serveru Kordisu, ale s tím rozdílem, že není využit HttpClient ale FtpClient.

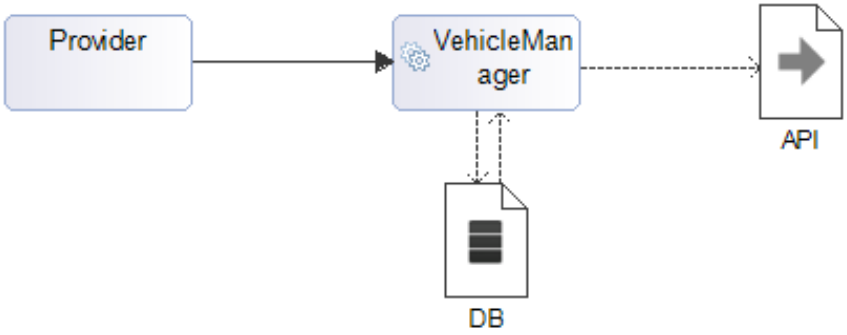
6.2.2 Poskytování dat

Abychom mohli zprostředkovávat získaná data, je potřeba tato data udržovat stále aktuální. Na začátku si můžeme rozdělit tato data do dvou skupin a to data s informacemi o vozidlech a data s informacemi o zastávkách.

6.2.3 Vozidla

Správu dat s vozidly má na starosti tzv. VehicleManager. Ten se stará jak o získávání dat ze systému poskytovatele, tak i následné ukládání do databáze a získávání zpět z databáze. Tento manager poskytuje jen obecné metody, které platí pro všechny ostatní managery starající se o vozidla. V aktuálním stavu existují další dva managery starající se o vozidla. Je to KordisVehicleManager, ZlinVehicleManager a pro případné další poskytovatele je nutno naimplementovat také vlastní manager. Tyto managery

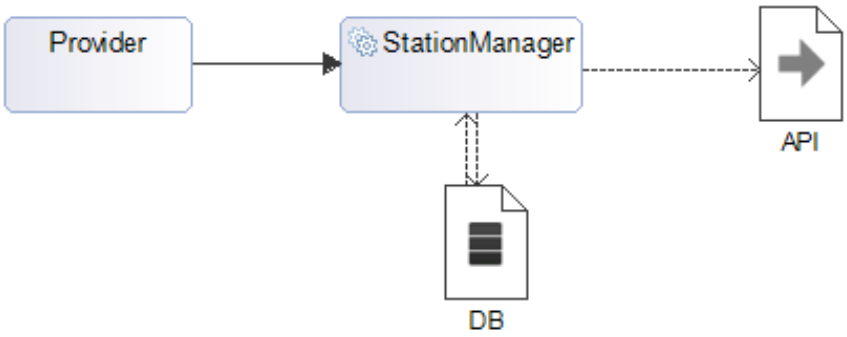
dědí všechny funkce poskytované obecným VehicleManagerem a navíc musí provádět konverzi dat na model, který poskytuje API.



Obr. 6.2 Proces diagram Vozidel

6.2.4 Zastávky

V případě zastávek je tomu podobně jako v předchozím případě. Existuje obecný StationManager, který pracuje hlavně s databází, ukládá do ní data o zastávkách a případně je aktualizuje nebo získává pro potřeby API. Stejně jako v předchozím případě existují zde třídy ZlinStationManager a KordisStationManager starající se o konverzi dat a ukládání do databáze.

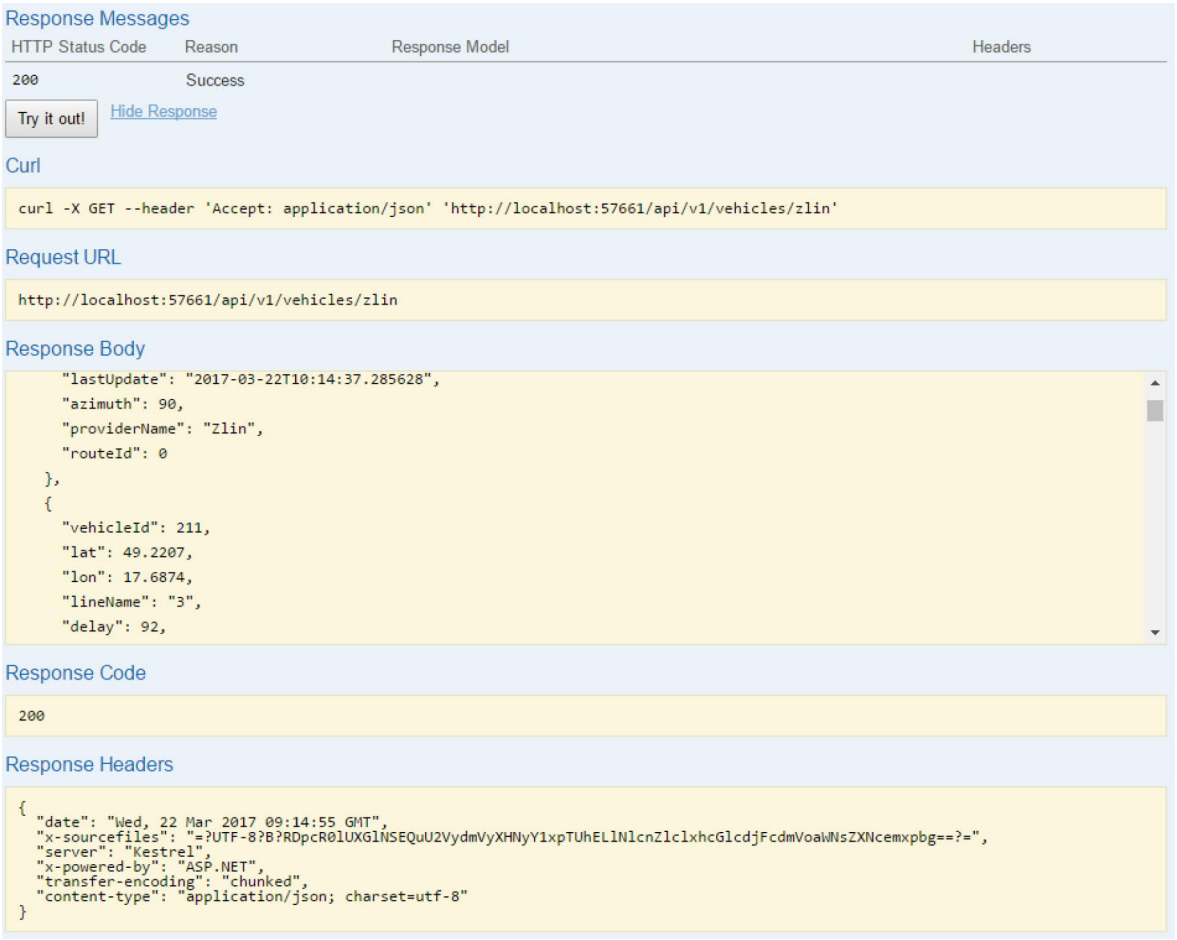


Obr. 6.3 Proces diagram Zastávek

O poskytování dat skrze API se starají controllery. Jejich úkol je celkem jednoduchý. Například při získávání vozidel ve Zlínském kraji se využije již zmiňovaný ZlinVehicleManager. Ten získá potřebná data jako seznam vozidel. Pokud vše dopadne v pořádku a podaří se data získat, musíme vozidla deserializovat. O to se postará právě controller, který současně s obsahem ve formátu JSON nastavuje i HTTP hlavičky.

```
[HttpGet]
[Route("api/v1/vehicles/zlin/")]
public async Task<IActionResult> GetZlinVehicles()
{
    IEnumerable<Vehicle> vehicleList;
    try
    {
        vehicleList = await _zlinVehicleManager.GetVehicles();
    }
    catch (Exception exception)
    {
        return BadRequest(exception.Message);
    }
    return Ok(vehicleList);
}
```

Tuto funkcionalitu lze vyzkoušet, aniž bychom museli mít mobilní aplikaci. Díky nástroji zvaném swagger, který generuje dokumentaci k API, můžeme vyzkoušet, že námi naprogramované funkce fungují.



Obr. 6.4 Swagger vozidla Zlín

6.2.5 Aktualizace dat

I přesto, že zastávky jsou uloženy lokálně v mobilní aplikaci, bylo potřeba vytvořit controller starající se o zastávky. Vznikl proto, že může nastat situace, kdy vznikne nová zastávka, případně v lokálním úložišti může chybět.

Data jsou vybírána z databáze a následně převedena na model, který je dále poskytován jako výstup tohoto controlleru. Aby byla zajištěna aktuálnost dat, je potřeba data periodicky aktualizovat. To je však potřeba provádět v různém čase, protože každý zdroj dat je aktualizován v jiném intervalu. Proto byl využit balíček s názvem "RecurrentTasks", který zajišťuje spouštění aktualizace dat v různém čase a různém intervalu.

Kdy a jak často se mají tyto procesy provádět, nastavíme opět ve Startup.cs pomocí následujícího kódu

```
//Zlin tasks
app.StartTask<ZlinVehicleUpdateTask>(TimeSpan.FromSeconds(30));
app.StartTask<ZlinStationUpdateTask>(TimeSpan.FromDays(1));
//Kordis tasks
app.StartTask<KordisVehicleUpdateTask>(TimeSpan.FromSeconds(5));
app.StartTask<KordisStationUpdateTask>(TimeSpan.FromDays(1));
```

Pro ilustraci nám bude stačit uvést jeden z úkolů, který se má opakovaně provádět v nějakém intervalu, abychom pochopili jeho funkci.

```
public class KordisStationUpdateTask : IRunnable
{
    private KordisStationManager _manager;
    public KordisStationUpdateTask(
            DbContextOptions<MhdDbContext> options,
            KordisFtpClient client)
    {
        var context = new MhdDbContext(options);
        _manager = new KordisStationManager(context, client);
    }
    public async void Run(ITask currentTask)
    {
        try
        {
            await _manager.UpdateStations();
        }
        catch (Exception e)
        {
            //TODO: log exception
        }
    }
}
```

Jak můžeme vidět, naše třída, která provádí nějakou činnost, implementuje interface IRunnable. Tento interface nám zajišťuje to, že třída bude mít metodu Run s parametrem ITask. Tato metoda se spustí asynchronně, tedy nečeká se na její výstup. Konkrétně v tomto případě se aktualizují stanice v databázi. Ostatní úlohy však fungují naprosto stejně, takže není potřeba je zde uvádět a vysvětlovat.

7 Mobilní aplikace

7.1 Potřebné nástroje

Vývoj multiplatformních aplikací by se neobešel bez vývojových nástrojů jako je Visual Studio, Xamarin Studio, emulátorů a testovacího prostředí. Pro každou platformu jsou potřeba jiné nástroje. Nejprve je však potřeba rozhodnout se, pro jaké platformy chceme vyvíjet.

	MACOS	WINDOWS	
Development Environment	XAMARIN STUDIO	VISUAL STUDIO	XAMARIN STUDIO
Xamarin.iOS	Yes	Yes (with Mac computer)	No
Xamarin.Android	Yes	Yes	Yes
Xamarin.Forms	iOS & Android only	Android, Windows, Windows Phone (iOS with Mac computer)	Android only
Xamarin.Mac	Yes	Open project & compile only ^	No

Obr. 7.1 Xamarin přehled podpory OS [1]

Pokud bychom se rozhodli pro vývoj především pro platformu iOS, Mac případně i Android, je rozumnější vyvíjet na operačním systému Mac OS. Naopak pokud budeme cílit primárně na Android nebo UWP, je výhodnější a v případě UWP dokonce i nutné zvolit operační systém Windows, který sice podporuje i sestavování balíků pro iOS, nicméně i tak je potřeba mít k dipozici další stroj s Mac OS a Mac Agentem. Operační systém Linux bohužel není Xamarinem podporován.

7.1.1 Android

Ať zvolíme jakýkoliv operační systém, v případě vývoje pro tuto platformu je nutné mít nainstalované nástroje

- Java JDK
- Android SDK
- Android NDK

Samozřejmě je nutné i zařízení pro testování aplikací, ať už reálné zařízení se systémem Android nebo emulátor. V případě volby emulátoru máme opět více možností. Pokud budeme používat IDE Visual Studio, je možné využít emulátoru, který je jeho součástí a využívá virtualizační technologii Hyper-V. Funguje opravdu velmi dobře a

je poměrně dost rychlý, otázka však je, jak moc se blíží skutečnému zařízení. Další možností je využít Android emulator pomocí nástroje AVD Manager od společnosti Google, který je ale velmi pomalý bez hardwarové akcelerace. Samozřejmě existují i další emulátory, které zde nebyly jmenovány.

Tak jako tak naše aplikace vyžaduje tzv. služby Google Play, které by v případě reálného zařízení měly být již předinstalované. V případě emulátoru tomu tak není a je potřeba je doinstalovat. Balíček s Google Apps lze stáhnout například ze stránek http://www.teamandroid.com/gapps/. Je však důležité zvolit správnou verzi, aby vše fungovalo správně.

7.1.2 iOS

Vývoj pro platformu iOS vyžaduje minimálně operační systém Mac OS ve verzi 10.11 a novější, abychom mohli vůbec Xamarin nainstalovat. Další nedílnou součástí je aplikace Xcode, kterou lze stáhnout přímo z App Store aktuálně ve verzi 8. Díky XCode můžeme spouštět naše aplikace na tzv. simulátoru, což je vlastně emulátor podobně jako je tomu na platformě Android.

7.1.3 UWP

Platforma UWP je velmi zajímavá svým způsobem vývoje. Pro vývoj této platformy je vyžadován systém Windows 8.1 a novější, zároveň je vyžadováno Windows 10 SDK. Nejvhodnějším nástrojem je Visual Studio, které nám umožňuje velmi pohodlně vyvíjet a sledovat využité prostředky. Díky tomu, že UWP aplikace jsou kompatibilní napříč zařízeními, není potřeba emulátor, pokud nevyužíváme speciálních funkcí jako je třeba posílání SMS. Nicméně v našem případě je vhodnější jej použít, protože právě mobilní zařízení jsou našimi cílovými zařízeními.

7.2 Implementace mobilní aplikace

Mobilní aplikace využívá zmíněný multiplatformní framework Xamarin.Forms, díky kterému je možné naprogramovat jedinou logiku a využít ji na všech podporovaných zařízeních.

7.3 Popis chování

Protože aplikace byla navržena tak, aby uměla konzumovat více poskytovatelů dat, byla zde zavedena logika, tzv. "Synchronizer", který se stará o to, jaký poskytovatel dat nebo "Manager"bude použit na základě pozice uživatele. Momentálně jsou využity pouze dva segmenty, takže se kontroluje pouze vzdálenost k nejbližšímu městu. Do budoucna je však vhodné tuto logiku rozšířit, rozdělit oblast do segmentů a určit, ve

kterém z těchto segmentů se uživatel nalézá. Problém by však mohl nastat, pokud se dva segmenty budou překrývat. Zmiňovaný Synchronizer garantuje to, že se zavede správný Manager a ten již načítá jen data pro danou oblast.

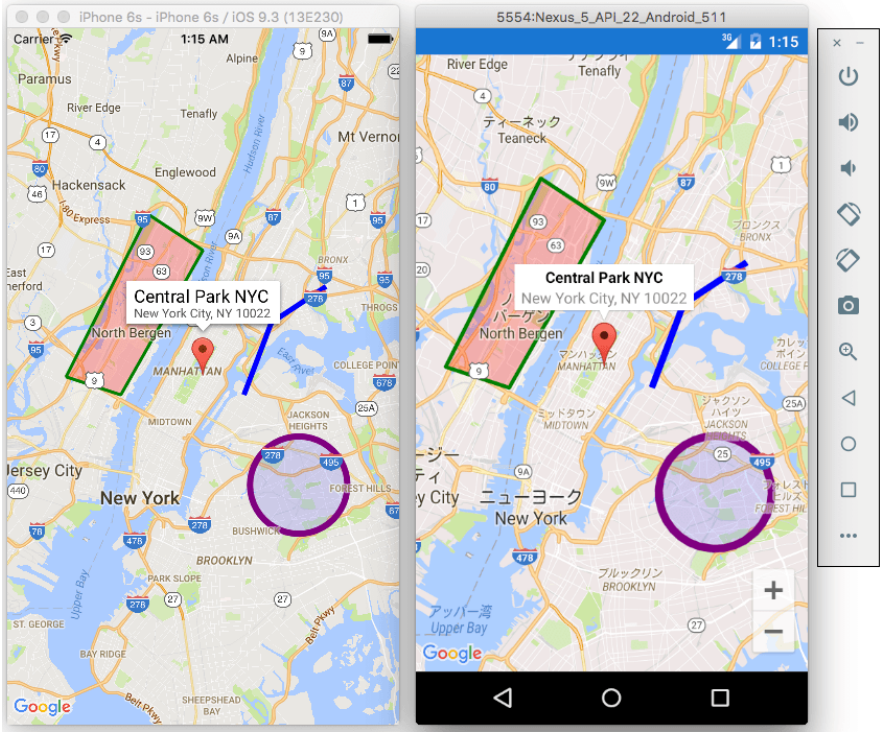
Data jsou aktualizována periodicky v intervalu 5 vteřin od spuštění aplikace. V případě že aplikace není aktivní, tedy spí nebo je ukončena, je i periodická aktualizace ukončena.

Stažená data jsou ukládána do lokálního cache úložiště. V případě, že daný záznam již v cache existuje, bude přepsán novými daty, v opačném případě se vytvoří nový. V obou případech má každý záznam určitou platnost a po uplynutí této doby je automaticky vymazán z cache úložiště.

Komponenta zobrazující data je vždy při aktualizaci dat upozorněna, že došlo k aktualizaci dat. Při aktualizaci dat je nutné překreslit objekty, které byly změněny. Ideálně však jen ty, které jsou aktuálně zobrazovány.

7.4 Vykreslování vozidel

Komponenta, která vykresluje mapu, umožňuje zároveň vkládat nejrůznější objekty do mapy jako jsou například body zájmu, linky, plochy a další. V našem případě to budou body zájmu, které budou představovat naše vozidla. Ty jsou vyznačeny na obrázku 7.2



Obr. 7.2 Xamarin Forms Google Maps [39]

Grafická reprezentace bodů zájmů sice umožňuje mnoho věcí, v našem případě je nedostačující. Uživatel není schopen vyčíst, jakým směrem je bod natočen. Zároveň, aniž bychom klikli na konkrétní bod, nevíme o něm další informace jako například číslo linky.

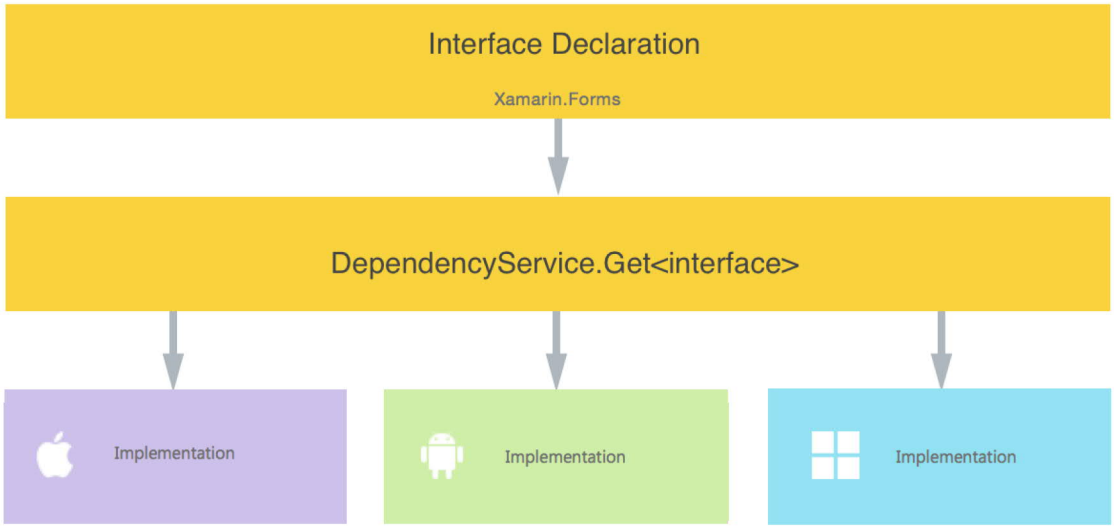
Zpočátku se zdálo jako nejlepší řešení pomocí SVG a XAML vytvořit objekt, který lze velmi jednoduše měnit. Vkládání textu, změna barvy a velikosti bylo opravdu jednoduché a nenáročné. Tento objekt se pokaždé musí transformovat jako obrázek a ten teprve lze zobrazit na mapě. S přibývajícím počtem objektů přestávalo toto řešení být efektivní a vyžadovalo velkou část výkonu při transformaci.

Další možností je vytvořit obrázky o různých velikostech a připojit je jako zdroje do balíku aplikace. Toto řešení již nevyžaduje tolik výpočetního výkonu a zdá se být schůdnější. Protože potřebujeme vozidlo jednoduše identifikovat, vykreslování názvu linky se zdá být jako vhodné řešení. I v této chvíli se však nabízí další možnosti. Jedna z možností je ukládat upravené obrázky již s textem do zařízení nebo je převést na datový typ Stream a s tím dále pracovat. Při ukládání obrázku do zařízení občas docházelo k tomu, že obrázek, ke kterému přistupovalo jedno vlákno, přistupovalo zároveň i další vlákno a nastávaly výjimky. Proto byla zvolena druhá možnost, kdy se pracuje pouze se streamy.

Přesto, že existují multiplatformní knihovny, které umí pracovat s obrázky, jejich použití je poměrně komplikované.

```
using System.IO;
namespace iMHD.Contract.Interface
{
    public interface IAssetsManager
    {
        Stream DrawToBitmap(string text, float degrees);
        string ReadFileFromAsset(string file);
    }
}
```

Implementace jednotlivých tříd na každé platformě je odlišná. Abychom byli schopni volat funkcionalitu s rozdílnou funkčností na každé platformě, bylo využito služby, kterou nabízí Xamarin tzv DependencyService.



Obr. 7.3 Dependency service [1]

DependencyService umožňuje aplikacím volání specifických funkcí pro danou platformu ze sdíleného kódu. Tato funkcionalita je součástí knihovny Xamarin.Forms a pomocí ní můžeme využívat nativní přístup aplikace.

DependencyService hledá správnou implementaci na dané platformě podle předloženého interface. Aby tato služba mohla fungovat, vyžaduje tyto tři komponenty

- Interface - požadovaná funkcionalita je specifikovaná v něm
- Implementation pro každou platformu – Třída, která implementuje interface, musí být přítomna v každém projektu pro každou platformu.
- Registrace – Každá implementace třídy musí být zaregistrována. Registrace umožňuje najít implementace třídy.

Registrace třídy je na všech platformách stejná, nicméně v případě platformy UWP je potřeba provést navíc i předání všech našich assembly do Xamarin Forms Inicializace.

```
[assembly: Xamarin.Forms.Dependency(typeof(AssetsManager))]
```

7.4.1 Implementace Android

Implementace pro platformu Android se zdá být nejjednodušší ze všech platforem. Stačí vytvořit obrázek v nejvyšším možném rozlišení, které chceme podporovat a Android si jej sám zmenší pro další rozlišení. Dále je nutné načíst obrázek jako bitmapu a

zkopírovat ji do nového objektu, se kterým můžeme dále manipulovat. Pak už stačí vytvořit plátno ve velikosti obrázku a správně vypočítat pozicování textu podle jeho velikosti.

```
public Stream DrawToBitmap(string text, float degrees)
{
    Resources resources = Application.Context.Resources;
    float scale = resources.DisplayMetrics.Density;
    var resourceList = new List<int>
    {
        Resource.Drawable.ic_arrow_blue
        //Seznam obrázků
    };
    var index = Convert.ToInt32(text);
    var resource = resourceList[index % resourceList.Count];
    //Vytvoření kopie bitmapy z obrázku
    Bitmap bitmap = BitmapFactory.DecodeResource(resources, resource);
    bitmap = bitmap.Copy(Bitmap.Config.Argb8888, true);
    //Kreslící plátno
    Canvas canvas = new Canvas(bitmap);
    canvas.Save();
    Rect bounds = new Rect();
    Paint paint = new Paint();
    paint.AntiAlias = true;
    paint.SubpixelText = true;
    paint.SetTypeface(Typeface.DefaultBold);
    paint.Color = Color.White;
    if (text.Length == 1)
    {
        paint.TextSize = 10 * scale;
    }
    else if (text.Length == 2)
    {
        paint.TextSize = 9 * scale;
    }
    else if (text.Length > 2)
    {
        paint.TextSize = 8 * scale;
    }
    paint.FakeBoldText = true;
    paint.GetTextBounds(text, 0, text.Length, bounds);
    float x = (bitmap.Width / 2) - (bounds.Width() / 2);
    float y = (bitmap.Height / 2) + (bounds.Height() / 2);
    //Vykreslení textu do obrázku
    canvas.DrawText(text, x, y, paint);
    canvas.Restore();
    MemoryStream stream = new MemoryStream();
    stream.Position = 0;
    bitmap.Compress(Bitmap.CompressFormat.Png, 0, stream);
    return stream;
}
}
```

7.4.2 Implementace iOS

V případě iOS je potřeba nejprve vytvořit obrázek ve velkém rozlišení a stejně tak ho vytvořit i pro menší rozlišení. Abychom mohli správně napozicovat text, je nutné

prvně nastavit kreslící mód jako neviditelný, vykreslit text a teprve poté spočítat jeho velikost, přepočítat velikosti písma a pozice a znovu již viditelně kreslit.

```
public Stream DrawToBitmap(string text, float degrees)
{
    List<string> pinList = new List<string>
    {
        "ic_arrow_blue.png"
        //Seznam obrázků
    };
    var index = Convert.ToInt32(text);
    var resource = pinList[index % pinList.Count];
    UIImage img = UIImage.FromBundle(resource);
    nfloat fWidth = img.Size.Width;
    nfloat fHeight = img.Size.Height;
    UIColor color = UIColor.White;
    using (CGBitmapContext ctx = new CGBitmapContext(
            IntPtr.Zero, (nint)fWidth, (nint)fHeight, 8, 4 * (nint)fWidth,
            CGColorSpace.CreateDeviceRGB(),
            CGImageAlphaInfo.PremultipliedFirst))
    {
        ctx.DrawImage(
            new CGRect(0, 0, (double)fWidth, (double)fHeight), img.CGImage);
        int fontHeight = 12;
        if (text.Length > 2)
        {
            fontHeight = 10;
        }
        ctx.SelectFont("Helvetica", fontHeight, CGTextEncoding.MacRoman);
        //Začátek měření délky textu
        float start, end, textWidth;
        //Získání pozice
        start = (float)ctx.TextPosition.X;
        //Kreslední neviditelnou barvou
        ctx.SetTextDrawingMode(CGTextDrawingMode.Invisible);
        //Draw the text at the current position
        ctx.ShowText(text);
        end = (float)ctx.TextPosition.X;
        //Konec měření délky textu
        textWidth = end - start;
        nfloat fRed;
        nfloat fGreen;
        nfloat fBlue;
        nfloat fAlpha;
        //Nastavení barvy textu
        color.GetRGBA(out fRed, out fGreen, out fBlue, out fAlpha);
        ctx.SetFillColor(fRed, fGreen, fBlue, fAlpha);
        //Set the drawing mode back to something that will actually draw Fill for
        ctx.SetTextDrawingMode(CGTextDrawingMode.Fill);
        var pointx = (fWidth / 2) - (textWidth / 2);
        var pointy = (fHeight / 2) - (fontHeight / 2);
        //Vykreslení textu na dané pozici
        ctx.ShowTextAtPoint(pointx, pointy, text);
        var image = UIImage.FromImage(ctx.ToImage());
        return image.AsPNG().AsStream();
    }
}
```

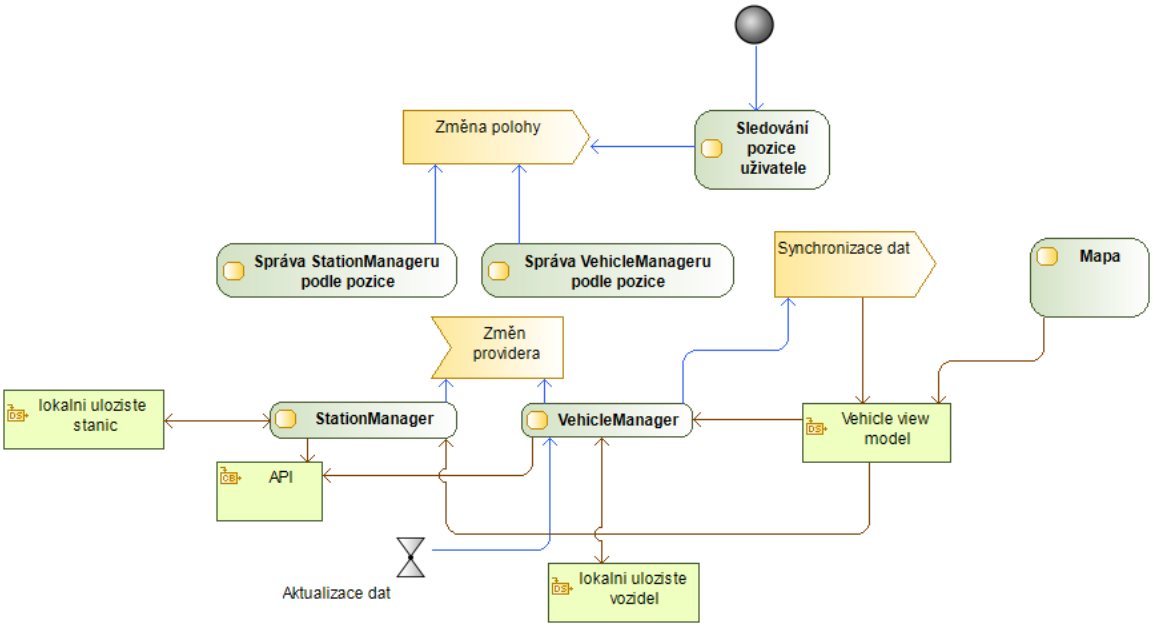
7.4.3 Implementace UWP

Implementace v případě platformy UWP není zcela hotová. Bing mapy totiž neumožňují otáčení objektů ve směru, tak jako tomu je u Google map a pokud je potřeba otočit objektem, musí se otočit obrázek celý. Sdílená část kódů momentálně nepočítá s touto indispozicí a je tedy potřeba při dalším rozšiřování projektu myslet na to, že pokud se změní azimut vozidla, je nutné překreslit i obrázek ve směru. Podpora knihoven třetích stran na platformy UWP není tak rozsáhlá jako v případě platformy Android a iOS, navíc chybí návody, jak pracovat s obrázky a kreslení na nich. Momentální implementace pouze získává obrázek z balíku a předává ho dále komponentě k vykreslení.

```
public Stream DrawToBitmap(string text, float degrees)
{
    List<string> pinList = new List<string>
    {
        "ic_arrow_blue.png",
         // Seznam obrázků
    };
    var index = Convert.ToInt32(text);
    var resource = pinList[index % pinList.Count];
    //Získávání obrázků z aplikace
    var file = StorageFile.GetFileFromApplicationUriAsync(
        new Uri($"ms-appx:///Assets/Markers/{resource}"))
        .AsTask()
        .ConfigureAwait(false)
        .GetAwaiter()
        .GetResult();
    var data = FileIO.ReadBufferAsync(file)
        .AsTask()
        .ConfigureAwait(false)
        .GetAwaiter()
        .GetResult();
    //Konverze obrázku na stream
    var ms = new InMemoryRandomAccessStream();
    var dw = new Windows.Storage.Streams.DataWriter(ms);
    dw.WriteBuffer(data);
    dw.StoreAsync().AsTask().ConfigureAwait(false).GetAwaiter().GetResult();
    ms.Seek(0);
    return ms.AsStream();
}
```

7.5 Diagramy

Na tomto diagramu je zobrazeno, jak spolu komunikují jednotlivé komponenty. Největší část logiky tvoří komponenty StationManager a VehicleManager, které obnovují data z API a ukládají do lokální cache. Dále je zapojen tzv. Vehicle view model, který přidává a odebírá vykreslované body do komponenty Mapa.



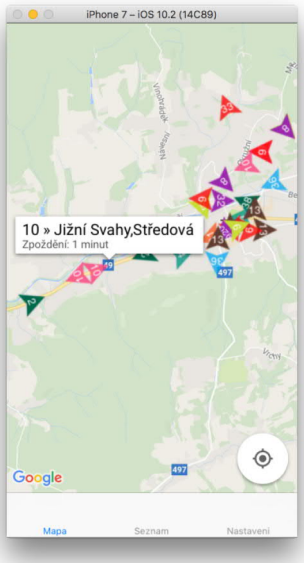
Obr. 7.4 Activity diagram Aplikace

7.6 Uživatelské rozhraní

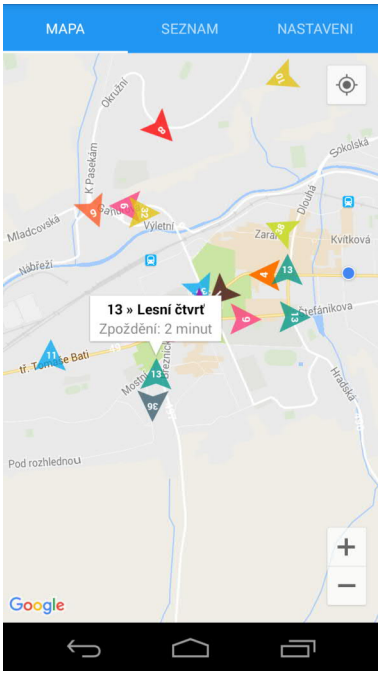
Aby aplikace byla co nejefektivnější, tedy aby uživatel viděl přesně to, co od aplikace očekává, bylo dbáno na to, aby byla vytvořena v co nejjednodušším provedení. Momentálně se skládá z karetního pásu se třemi kartami. Karta s mapou, která bude zobrazována nejčastěji, je na prvním místě. Na platformě Android a iOS je k dispozici tlačítko, které lokalizuje polohu uživatele na mapě. Platforma UWP bohužel tuto funkcionalitu momentálně nemá. Společně ale všechny platformy umí zobrazit polohu uživatele jako modrou tečku.

Další karta zobrazuje seznam všech dostupných vozidel a informace o nich. Tato karta pomohla při ladění aplikace, zároveň dokáže být velmi užitečná. Například v případě, že bychom se domluvili s někým na společné jízdě stejným spojem a spoj by nejel zrovna na čas, stačí, když nám dotyčný sdělí číslo spoje a pak jsme schopni jej sledovat na mapě.

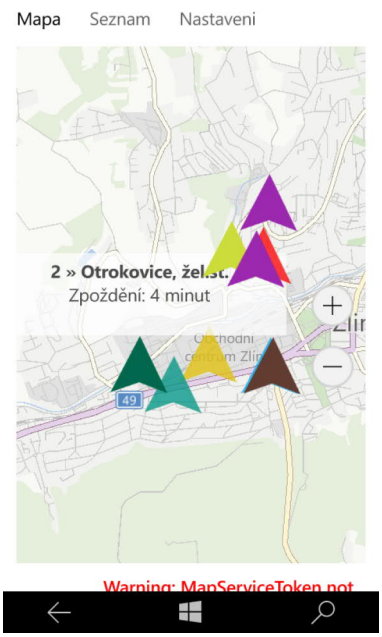
Třetí a poslední karta umožňuje nastavení aplikace. Momentálně má tři možnosti nastavení. První z nich je doprava, která nám zobrazuje stav dopravy na mapě vyznačená zelenou barvou v případě slabého provozu, žlutá barva pro střední provoz a červená barva pro velký provoz, kdy můžou nastávat i kolony. Dalším nastavením je zobrazování polohy uživatele. Pokud by uživatele nezajímala jeho poloha a byla pro něj zbytečná, je možné ji vypnout tlačítkem zobrazit polohu. Posledním natavením je sledování polohy. Tato funkce momentálně není implementována, ale je zamýšlena jako sledování polohy uživatele. Tedy pokud se změní pozice uživatele, pozice tzv. kamery se přemístí na místo, kde se uživatel momentálně nalézá a vycentruje se.



Obr. 7.5 Mapa vozidel iOS

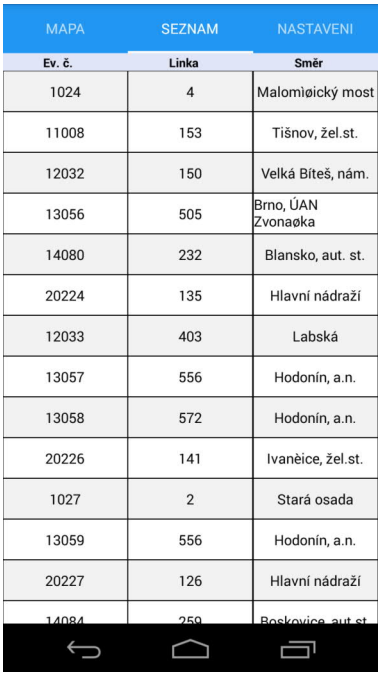


Obr. 7.6 Mapa vozidel Android



Obr. 7.7 Mapa vozidel UWP

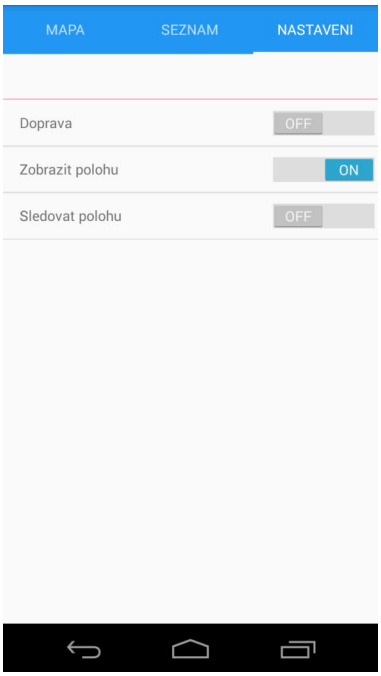
Obr. 7.8 Seznam vozidel iOS



Obr. 7.9 Seznam vozidel Android

Obr. 7.10 Seznam vozidel UWP

Obr. 7.11 Nastavení aplikace iOS



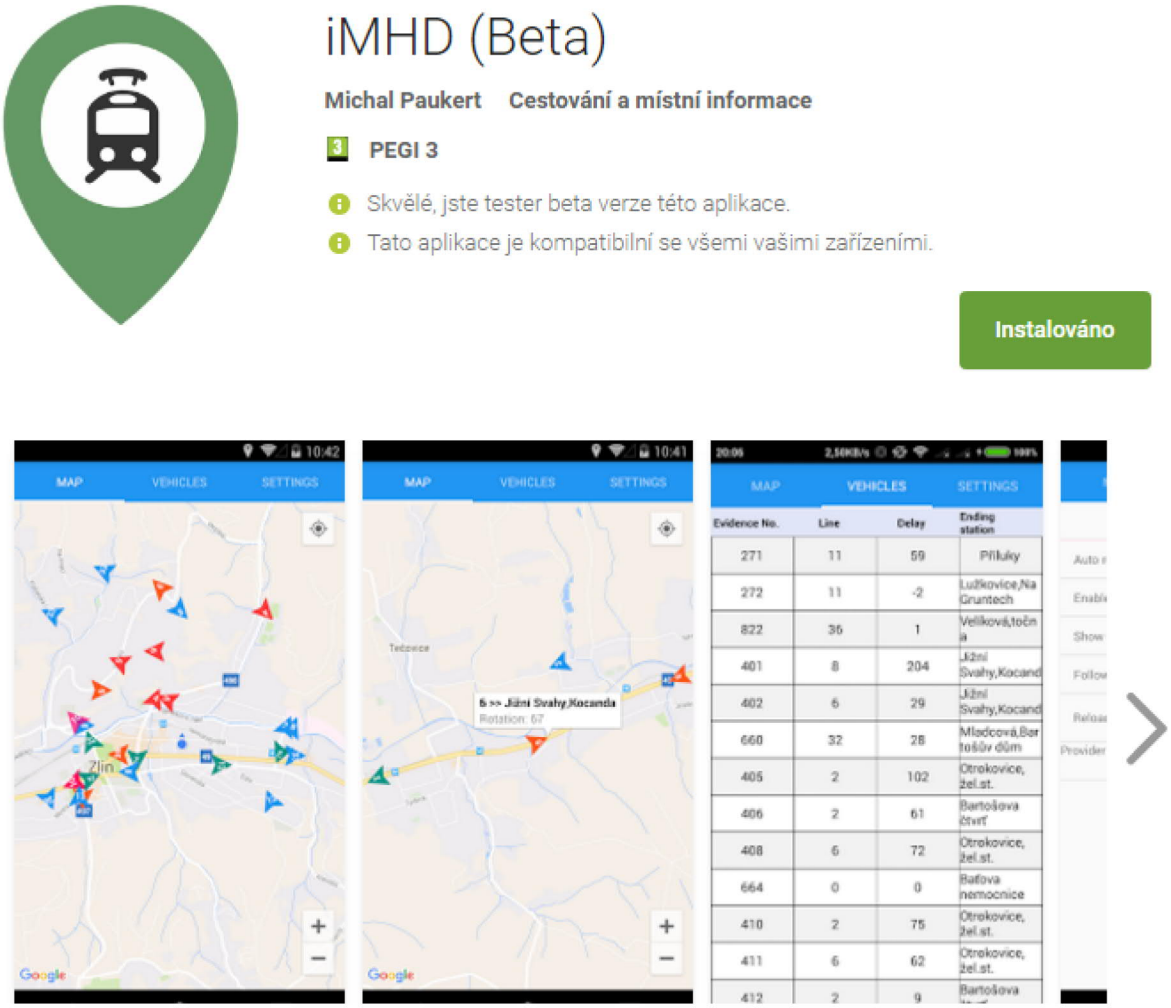
Obr. 7.12 Nastavení aplikace Android

Obr. 7.13 Nastavení aplikace UWP

7.7 Publikování aplikace

7.7.1 Android

Jeden ze způsobů, jak publikovat aplikaci, je využít oficiálního obchodu, což ale nebude úplně zadarmo. Nebo můžeme využít levnější možnost a to je zveřejnění instalačního balíčku. Obě zmíněné možnosti mají své výhody a nevýhody a budou popsány níže. Pro Android je k dispozici oficiální obchod s aplikacemi, který se nazývá Google Play. Díky němu lze distribuovat naše aplikace a je pak dostupná pro všechny majitele kompatibilních zařízení. [37] Abychom mohli tento obchod využívat, je nutné zaplatit jednorázový registrační poplatek v hodnotě $25 USD. Po zaplacení tohoto poplatku přejdeme do Google Developer Console, kde vytvoříme svoji aplikaci a vyplníme potřebné údaje o ni. Pro nahrání aplikace do obchodu můžeme využít buď Visual Studio, nebo můžeme balíček nahrát manuálně do obchodu. Pokud použijeme Visual studio, nemusíme se téměř o nic starat. Balíček, který se bude nahrávat, se vygeneruje sám spolu s certifikátem, který je nutné dobře uschovat. Bez zmíněného certifikátu již nebudeme schopni nahrát znovu naši aplikaci se stejným balíčkovým jménem. Poté přichází na řadu výběr kanálu distribuce. Na výběr máme Alfa a Beta kanál, který můžeme využít pro testování a produkční kanál, který uveřejní naši aplikaci v obchodě pro ostatní uživatele. Zveřejnění je delší proces a prochází určitým schvalováním, což může trvat i pár hodin.



Obr. 7.14 Android aplikace v Google Play

Pokud bychom chtěli využít první možnosti, tedy sdílení naší aplikace například na svých stránkách, je to možné. Na rozdíl od předešlé možnosti má tato volba spíše více nevýhod než výhod. Výhodou je, že neplatíme žádný registrační poplatek a naše aplikace nemusí být nikým revidována při publikaci. Nevýhodou je to, že se musíme starat o propagaci aplikace a instalace balíčku není tak jednoduchá jako v předchozím případě. Uživatel si musí nahrát balíček do telefonu a poté nainstalovat. Navíc zde není kontrola podporovaných zařízení.

7.7.2 UWP

Oficiální obchod pro Windows má název Microsoft Store. Pokud bychom chtěli publikovat naši aplikaci v tomto obchodě, musíme zaplatit taktéž registrační poplatek a záleží na tom, jestli nám jde o individuální vývoj nebo o vývoj pro firemní účely, kde Microsoft nabízí více možností. V případě individuálního vývoje zaplatíme jednorázově

$19 USD.

V případě firemního účtu je poplatek o něco větší, přesněji $99 USD. Je to hlavně proto, že je nutné provést některé další kroky k zajištění toho, že uživatel, který nahrává aplikaci, má oprávněni zastupovat firmu a spravovat účet.

Postup pro publikaci v tomto obchodě je velmi podobný jako v případě Google Play. Doporučený postup je nahrát pomocí Visual Studia aplikaci do obchodu, doplnit údaje o aplikaci a předat k revizi. Následně prochází zdlouhavým procesem, kde dochází ke kontrole aplikace z pohledu bezpečnosti, jestli neobsahuje malware nebo není nějak jinak napadena. Dále je testována pomocí nástroje "Windows App Certification Kit", který obsahuje velké množství testů k ověření aplikace. Zbytek schvalovacího času závisí na tom, jak komplexní je naše aplikace po vizuální stránce.

7.7.3 iOS

Oficiálním obchodem pro platformu iOS je App Store. Pokud chceme šířit naše aplikace veřejně, je to v podstatě jediný vhodný způsob, jak toho docílit. Apple má totiž poměrně striktní pravidla a nedovolí instalovat neověřené aplikace. Abychom mohli nahrávat aplikace do App Store, musíme si opatřit vývojářskou licenci, která momentálně stojí $99 USD a na rozdíl od předchozích se platí ročně. Díky této licenci je možné se zařadit do iOS developer programu a stát se tak platným iOS vývojářem. Při registraci naší aplikace do obchodu získáme certifikát, kterým je nutné podepsat naši aplikaci, aby mohla být zveřejněna.

7.8 Testování mobilní aplikace

Pro testovací účely byl využit mobilní telefon Xiaomi Redmi Note s operačním systémem Android. Při testování bylo zjištěno, že aplikace funguje velmi dobře. V případě, že pozice uživatele byla nastavena ve městě Zlín a tedy dostupných objektů k zobrazování na mapě bylo mnohem méně, aplikace působila svižným dojmem. V případě, že pozice uživatele byla umístěna v Jihomoravském kraji, tedy blíže k městu Brnu, kde těchto objektů bylo mnohem více, výkon telefonu přestával být dostačující, pokud bylo potřeba vykreslit velké množství objektů. K tomuto by nemuselo docházet, pokud uživatel bude udržovat přiblížení na svoji pozici a její okolí, nikoliv na celé město.

Jako další testovací zařízení bylo využito Apple iPhone 6S s operačním systémem iOS. Aplikace na tomto zařízení má občas problémy s výkonem a dochází k „zamrznutí“ obrazovky na delší dobu. Pro reálné využití a příjemnému uživatelskému zážitku je zapotřebí se zaměřit na rychlost této aplikace pro tuto platformu.

Reálné mobilní zařízení s Windows Phone se nepodařilo získat v průběhu vývoje této práce a nebylo tedy možné ji otestovat. Nicméně pokud nedojde k zásadním změnám

u této platformy, bude velmi složité podporovat vývoj pro tuto platformu a vyvíjet funkce, které jsou k dispozici pro iOS a Android.

7.8.1 Rozdíly platforem

I přesto, že aplikace je navržena pro všechny tři platformy stejně, jsou zde menší rozdíly. Karty jsou v případě iOS umístěny ve spodní části aplikace, u Androidu jsou zase v horní části s modrým podbarvením a v případě UWP jsou taktéž v horní části, ale bez podbarvení. Stejně tak i tlačítka vypadají tak, jako by byla systémová. To je způsobeno tím, jak jsou platformy navrženy a jejími pravidly pro návrhu UI, jak mají jednotlivé komponenty vypadat.

7.9 Návrhy na zlepšení

7.9.1 Filtrování dat

Jak již bylo zmíněno při testování, je vhodné rozlišit typy vozidel a linek a na tomto základě filtrovat vozidla. Tedy vhodné vlastnosti podle kterých je možné filtrovat jsou

- Typ vozidla
- Číslo linky
- Pouze nízkopodlažní vozy

7.9.2 Optimalizace aplikace

Vzhledem k cenám mobilních tarifů a velikosti FUP v ČR je více než vhodné optimalizovat velikost dat, která jsou periodicky stahována. Když aplikace byla testována, jako připojení k internetu využívala Wi-Fi, takže velikost stažených dat nebyla podstatná. Nicméně toto považuji jako další vhodné rozšíření.

7.9.3 Další poskytovatelé dat

V rámci této práce byl stanoven cíl zpracovávat alespoň dva zdroje dat a je tedy splněn získáváním dat od dopravců Kordis a DSZO. Tento cíl byl stanoven kvůli tomu, abychom dokázali, že je možné zpracovávat více zdrojů a poskytovat je pro mobilní zařízení. To nám však nebrání ve spolupráci s dalšími městy. Z důvodu časové náročnosti se nebudeme dále zabývat dalšími zdroji. Nicméně je to velice vhodné jako další rozšíření této práce.

7.9.4 Překlady

Přesto, že aplikace je zaměřena na spoje v České republice, není vyloučeno, že ji nebudou využívat i cizinci. Z tohoto důvodu bychom mohli doporučit aplikaci přeložit. Pro nastavení správného překladu můžeme využít jazyk nastavený v systému telefonu. Takto bychom mohli celkem spolehlivě poznat jazyk uživatele.

7.9.5 Více uživatelských nastavení

Ze začátku implementace byla snaha povolit uživateli co největší svobodu a nastavovat tak téměř vše, co lze nastavit. To se ze začátku zdálo jako dobrá volba, nicméně s přibývajícím počtem nastavení aplikace přestávala být přehledná. Snahou této aplikace je zůstat co nejjednodušší, a tak bylo rozhodnuto, že na základě uživatelské zpětné vazby se nastaví hodnoty, tak aby vyhovovaly co největšímu procentu uživatelů využívajících tuto aplikaci. Hodnoty, které momentálně nelze změnit, mohou být dopočítávány na základě nějaké skutečnosti. Je zde však prostor na analýzu ideálního nastavení, to však nelze provést, dokud nebudeme mít dostatečně velké množství uživatelů a výsledky by v tuto chvíli byly zkresleny.

7.9.6 Podrobnější data o vozech

Momentálně aplikace zobrazuje pouze základní informace o spojích jako je číslo linky a její směr. Přesto, že tyto informace jsou k dipozici, zůstávají nevyužity. Zejména pro vozíčkáře a cestující s kočárkem může být zajímavá informace o tom, zda je vůz nízkopodlažní.

ZÁVĚR

Tato diplomová práce měla jako hlavní cíle vypracovat literární studii na dané téma, analyzovat výběr možných technologií, vytvořit serverovou aplikaci jako agregátor dat, vytvořit mobilní aplikaci, která je konzumentem poskytovaných dat z tohoto serveru. V neposlední řadě i otestovat tuto aplikaci na reálném zařízení.

V rámci této práce byla provedena literární studie na dané téma a v teoretické části byly rozebrány existující aplikace se stejným zaměřením jako má tato diplomová práce. Popsány jsou jejich výhody, nevýhody a stručný popis. Dále byly popsány hlavní mobilní platformy a jejich procentuální zastoupení na trhu, pro které bylo rozhodnuto vyvíjet. Aby bylo možné rozhodnout, jakou technologii zvolit pro multiplatformní vývoj, bylo nutné provést analýzu vhodných technologií, respektive frameworků pro multiplatformní vývoj. Momentálně nejvhodnějším frameworkem pro tuto práci se stal Xamarin a i ten je v této práci popsán. Analytická část se zaměřuje na zdroje dat od třetích stran. Tato část popisuje, jakým způsobem jsou data získávána od poskytovatelů dat a význam těchto dat. Popsána je jak struktura dat o vozidlech, tak i o zastávkách, která je dále využita v projektu.

Projektová část se zabývá návrhem jak serverové aplikace, tak i implementací a vysvětlením některých hlavních částí, jak jsou data získávána a jak jsou poskytována pro mobilní aplikaci. Součástí projektové části je i postup přípravy prostředí pro vývoj mobilních aplikací pomocí frameworku Xamarin. Nejdůležitější částí je návrh a implementaci mobilní aplikace, tedy její hlavní části. Aplikace v rámci této práce se podařilo vytvořit pro všechny tři hlavní platformy a vytvořili jsme snímky jednotlivých obrazovek, jak ve skutečnosti aplikace vypadá. Na počátku se zdálo, že nejtěžší bude vývoj pro platformu Android, která nese pověst pomalého systému. Díky otevřenosti tohoto systému a velkému množství návodů se podařilo naimplementovat pro tuto platformu funkce jako první.

Funkce, které byly potřeba pro platformu iOS, již stavěly na zkušenostech z platformy Android a vývoj tak byl poměrně rychlý. I tak bylo potřeba překonat mnoho překážek. Například díky odlišnému chování systému povolit nezabezpečenou komunikaci pomocí http protokolu, nebo jiné rozmístění UI prvků.

Vývoj pro platformu UWP můžeme považovat za nejvíce komplikovaný. Vzhledem k chybějícím funkcím ze strany map bing maps bychom mohli označit tuto část jako částečně nekompletní, jak můžeme pozorovat na obrázku 7.10, kde nejsou vozidla natočena správným směrem.

I přesto však můžeme říci, že tato práce byla zdárně dokončena a splnila své stanovené cíle.

SEZNAM POUŽITÉ LITERATURY

[1] *Developer Center - Xamarin [online]. San Francisco, 2017 [cit. 2017-04-06]. Dostupné z:* https://developer.xamarin.com/

[2] *Chytré telefony zvyšují počet uživatelů internetu [online]. Praha, 2016 [cit. 2017-04-06]. Dostupné z:* https://www.czso.cz/csu/czso/chytre-telefony-zvysuji-pocet-uzivatelu-internetu

[3] *KLOS, Roman. Multiplatformní vývoj mobilních aplikací pomocí webových technologií [online]. Praha, 2014 [cit. 2017-04-06]. Dostupné z:* https://www.vse.cz/vskp/41206_multiplatformni_vyvoj_mobilnich_aplikaci_pomoci_webovych_technologii. *Diplomová práce. VŠE. Vedoucí práce Pavlíčková Jarmila.*

[4] *RUMPLI, Marko. Návrh a vývoj multiplatformních mobilních aplikací [online]. Hradec Králové, 2015 [cit. 2017-04-06]. Dostupné z:* https://theses.cz/id/y6ahyr/STAG83341.pdf

[5] *ČÁPEK, Petr. Moderní metody tvorby nativních multiplatformních mobilních aplikací [online]. Zlín, 2014 [cit. 2017-04-06]. Dostupné z:* https://digilib.k.utb.cz/handle/10563/30263

[6] *VRBKA, Tomáš. Moderní metody vývoje nativních mobilních aplikací s použitím Xamarin.Forms [online]. Zlín, 2016 [cit. 2017-04-06]. Dostupné z:* https://digilib.k.utb.cz/bitstream/handle/10563/38565/vrbka_2016_dp.zip

[7] *BOJANVSKÝ, Martin. Vývoj aplikací pro Universal Windows Platform [online]. Zlín, 2016 [cit. 2017-04-06]. Dostupné z:* http://digilib.k.utb.cz/bitstream/handle/10563/38794/bojnanský_2016_dp.zip

[8] *PINKAVA, Marek. Vývoj Multiplatformní Geolokační Aplikace [online]. Brno, 2012 [cit. 2017-04-06]. Dostupné z:* https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=57415

[9] *Jak náročné je získat data o poloze vozidel MHD? [online]. Brno [cit. 2017-04-06]. Dostupné z:* https://www.superlectures.com/openalt2014/downloadFile?id=2&type=slides&filename=jak-narocne-je-ziskat-data-o-poloze-vozidel-mhd

[10] *Podívejte se na vizualizaci historie zastoupení jednotlivých verzí Androidu. Svět Androida [online]. 2017 [cit. 2017-04-06]. Dostupné z:* https://www.svetandroida.cz/historie-zastoupeni-android-verze-201701

[11] *ZAPLETAL, Libor. Vývoj her pro mobilní platformy [online]. Brno, 2016 [cit. 2017-04-06]. Dostupné z:* https://is.muni.cz/th/430641/fi_m/diplomova-prace.pdf

[12] *FOJTÍK, David. INFORMAČNÍ SYSTÉM PRO SPRÁVU NEMOVITOSTÍ [online]. Brno, 2016 [cit. 2017-04-06]. Dostupné z:* https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=132290

[13] *KRČÍK, Michal. Multiplatformový vývoj mobilních aplikací [online]. Brno, 2014 [cit. 2017-04-06]. Dostupné z:* http://is.muni.cz/th/396551/fi_b/Bakalarska_praca.pdf

[14] *Jak Vyvíjet Mobilní Aplikace [online]. 2016 [cit. 2017-04-06]. Dostupné z:* http://janvaclavik.cz/jak-vyvijet-mobilni-aplikace/

[15] *LYSÁK, Adam. Systém pro vzdálený monitoring a konfiguraci škálovatelných služeb [online]. Praha, 2017 [cit. 2017-04-06]. Dostupné z:* https://dspace.cvut.cz/bitstream/handle/10467/66870/F3-DP-2017-Lysak-Adam-DIP_Lysak_Adam_2017.pdf?sequence=-1

[16] *Technologický přehled kolem ASP.NET Core 1.0 RC2 - Miroslav Holec [online]. 2016 [cit. 2017-04-06]. Dostupné z:* https://www.miroslavholec.cz/blog/technologicky-prehled-kolem-asp-net-core-1-0-rc-2

[17] *BENEŠ, Ondřej. NÁVRH INTERNETOVÝCH STRÁNEK [online]. 2011 [cit. 2017-04-06]. Dostupné z:* https://core.ac.uk/download/pdf/30285061.pdf

[18] *OLEJNÍK, Jakub. Florbalový trenér [online]. 2014 [cit. 2017-04-06]. Dostupné z:* https://dip.felk.cvut.cz/browse/pdfcache/olejnjak_2014bach.pdf

[19] *IDS JMK - Integrovaný dopravní systém Jihomoravského kraje včetně MHD Brno. IDS JMK - Integrovaný dopravní systém JMK - jízdní řády Brno a Jihomoravský kraj - oficiální stránky [online]. 2017 [cit. 2017-04-06]. Dostupné z:* http://www.idsjmk.cz/onas.aspx

[20] *MHD Brno » Vozový park » Řídící a informační systém brněnské MHD. MHD Brno [online]. c2002-2017 [cit. 2017-04-06]. Dostupné z:* http://www.bmhd.cz/evidence-dpmb/ris.php

[21] *Využití webových služeb a protokolu SOAP při komunikaci [online]. 2002 [cit. 2017-04-06]. Dostupné z:* http://www.kosek.cz/diplomka/html/websluzby.html

[22] *Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript. Movable Type — Information Design Management [online]. [cit. 2017-04-06]. Dostupné z:* http://www.movable-type.co.uk/scripts/latlong.html

[23] *Wifi Tram - Aplikace pro Android ve službě Google Play. Google Play [online]. [cit. 2017-04-06]. Dostupné z:* https://play.google.com/store/apps/details?id=cz.qwasar.wifitram&hl=cs

[24] *Wifitram RIS. RIS [online]. online [cit. 2017-04-30]. Dostupné z:* https://ris.azurewebsites.net/

[25] *iRIS - Aplikace pro Android ve službě Google Play. Google Play [online]. [cit. 2017-04-06]. Dostupné z:* https://play.google.com/store/apps/details?id=cz.zolex.iris&hl=cs

[26] *IRIS. IRIS - Poloha vozidel IDS JMK a MHD Brno [online]. online [cit. 2017-04-30]. Dostupné z:* https://iris.bmhd.cz

[27] *Šotoris - Aplikace v Microsoft Store. Oficiální domovská stránka Microsoft [online]. [cit. 2017-04-06]. Dostupné z:* https://www.microsoft.com/cs-cz/store/p/sotoris/9nblggh0d9pd

[28] *Šotoris - Blackberry World. BlackBerry App World [online]. 2014 [cit. 2017-04-06]. Dostupné z:* https://appworld.blackberry.com/webstore/content/59944042/?countrycode=CZ&lang=en

[29] *XProduction - Easy Brno. X Production Digital [online]. [cit. 2017-04-06]. Dostupné z:* http://www.xproduction.cz/cs/easy-brno-mobilni-aplikace

[30] *HejbejBrnem. HejbejBrnem [online]. [cit. 2017-04-06]. Dostupné z:* https://www.hejbejbrnem.cz/o-aplikaci

[31] *Mapa - Aktuální poloha přazských tramvají [online]. [cit. 2017-04-06]. Dostupné z:* https://tram.mobilnitabla.cz/

[32] *Dopravní společnost Zlín-Otrokovice, s.r.o. [online]. 2004 [cit. 2017-04-06]. Dostupné z:* http://www.dszo.cz/

[33] *DUDOVÁ, Veronika. Rozvrh hodin pro mobilní zařízení [online]. 2012 [cit. 2017-04-06]. Dostupné z:* http://www.itspy.cz/wp-content/uploads/2014/05/acmspy2012_submission_24.pdf

[34] *HUDEC, Jiří. Na příkladu demonstrujte vývoj aplikací pro systém Windows 10 [online]. 2015 [cit. 2017-04-06]. Dostupné z:* https://www.vse.cz/vskp/id/1268625

[35] *Using Apache web server as a reverse proxy. Docs.microsoft.com | Microsoft Docs [online]. 2016 [cit. 2017-04-06]. Dostupné z:* https://docs.microsoft.com/en-us/aspnet/core/publishing/apache-proxy

[36] *VRBA, Josef. Přehled webových mashupů [online]. Plzeň, 2012 [cit. 2017-04-06]. Dostupné z:* https://otik.uk.zcu.cz/handle/11025/3082

[37] *NOVÁK, Lukáš. Aplikace na objednávání jídel pro platformu Android [online]. 2012 [cit. 2017-04-06]. Dostupné z:* http://www.diplomovaprace.cz/2012/54/novak_bp_text.pdf

[38] *Schéma Centrální evidence dopravního řídícího střediska. In: Dotace z Evropské unie - ROP JV [online]. 2013 [cit. 2017-04-06]. Dostupné z:* http://www.jihovychod.cz/download/zakazky/560/p01_schema_ris_cedris.pdf

[39] *Map library for Xamarin.Forms using Google maps API [online]. 2013 [cit. 2017-04-06]. Dostupné z:* https://github.com/amay077/Xamarin.Forms.GoogleMaps

[40] *IDC: Smartphone OS Market Share 2016, 2015 [online]. 2016 [cit. 2017-04-06]. Dostupné z:* http://www.idc.com/promo/smartphone-market-share

[41] *Dashboards | Android Developers [online]. 2016 [cit. 2017-04-06]. Dostupné z:* https://developer.android.com/about/dashboards/index.html

[42] *App Store - Support - Apple Developer [online]. 2016 [cit. 2017-04-06]. Dostupné z:* https://developer.apple.com/support/app-store/

[43] *Nokia Lumia 520 is still the most popular Windows Phone Device | MobiTrends. In: MobiTrends [online]. 2016 [cit. 2017-04-06]. Dostupné z:* http://mobitrends.co.ke/windows-phone-market-share/.html

[44] *Vývoj aplikací pro Univerzální platformu Windows (UWP) [online]. 2016 [cit. 2017-04-06]. Dostupné z:* https://msdn.microsoft.com/cs-cz/library/dn975273.aspx

[45] *Chytsibus | MHD Bratislava [online]. 2017 [cit. 2017-04-06]. Dostupné z:* http://www.chytsibus.eu/

[46] *BŘEZINA, Lukáš. Vizualizace veřejné dopravy v Jihlavě [online]. Jihlava, 2012 [cit. 2017-04-06]. Dostupné z: https://is.vspj.cz/bp/getbp/student/27337/thema/2136*

[47] *NeverRun: už nikdy nezmeškejte svůj spoj MHD | Techbrain [online]. 2015 [cit. 2017-04-06]. Dostupné z:* `http://techbrain.cz/2015/06/15/neverrun-uz-nikdy-nezmeskejte-svuj-spoj-mhd/`

[48] *Wi-Fi v MHD, autobusech, tramvajích, trolejbusech | Wi-Fi v MHD [online]. [cit. 2017-04-06]. Dostupné z:* `http://www.wifivmhd.cz/wifi-v-mhd/`

[49] *Bratislava mohla mať mapu všetkých spojov MHD, no má mapu odhŕňačov [online]. 2016 [cit. 2017-04-06]. Dostupné z:* `https://dennikn.sk/336300/bratislava-mohla-mat-mapu-vsetkych-spojov-mhd-no-ma-mapu-odhrnacov/`

[50] *CityHack 2015 - sotoris.cz [online]. 2015 [cit. 2017-04-09]. Dostupné z:* `http://sotoris.cz/datasource/cityhack2015/`

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ČSÚ	Český Statistický Úřad
RIS	Řídící a informační systém
CED	Centrální Dispečink
CEDRIS	Centrální dispečink řídící a informační systém
DPMB	Dopravní podnik města Brna
MHD	Městská hromadná doprava
JMK	Jihomoravský kraj
GPS	Global Positioning System
WP	Windows Phone
WM	Windows Mobile
UWP	Universal Windows Phone
AVD	Android Virtual Device
WSDL	Web Services Description Language
XML	eXtensible Markup Language
SOAP	Simple Object Access Protocol
JSON	JavaScript Object Notation
CSV	Comma separated values
Wi-Fi	Wireless Fidelity
OS	Operační systém
SDK	Software development kit
NDK	Native Development Kit
API	Application Programming Interface
UI	User interface
HTML	HyperText Markup Language
PHP	Hypertext Preprocessor
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
CSS	Cascading Style Sheets
RSS	Rich Site Summary
MVC	Model view controller
CLI	Command Line Interface
SVG	Scalable Vector Graphics
XAML	Extensible Application Markup Language

SEZNAM OBRÁZKŮ

SEZNAM TABULEK