FAKULTA
ELEKTROTECHNICKÁ
ČVUT V PRAZE

JÍZDNÍ ŘÁDY S ODHADEM ZPOŽDĚNÍ POMOCÍ ZPĚTNÉ VAZBY UŽIVATELŮ

BAKALÁŘSKÁ PRÁCE

JAN KLÁN

PROGRAM: Otevřená informatika
OBOR: Software
VEDOUCÍ: Ing. Ivo Malý, Ph.D.

2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Klán** Jméno: **Jan** Osobní číslo: **466314**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Jízdní řády s odhadem zpoždění pomocí zpětné vazby uživatelů

Název bakalářské práce anglicky:

Pokyny pro vypracování:

Proveďte analýzu požadavků návštěvníků a občanů měst v oblasti dopravních informací. Zaměřte se na cestování různými druhy autobusové dopravy, záznam zpoždění a mimořádností. Srovnejte požadavky s již existujícími řešeními, jako je IDOS nebo CG Transit. Na základě analýzy navrhněte mobilní aplikaci pro sběr dat a jejich využití k přenějšímu odhadu aktuálního provozu. Uživatelské rozhraní otestujte ve formě nízkoúrovňových prototypů s alespoň 3 uživateli. Výslednou aplikaci implementujte na platformě Android a otestujte ji pomocí uživatelských a softwarových testů.

Seznam doporučené literatury:

[1] K. Boogaart, You, I, and ReactiveUI, 2018.
[2] M. Wallace, Android Development Patterns, Addison-Wesley Educational Publishers Inc, 2015.
[3] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ivo Malý, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **14.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Ivo Malý, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

PODĚKOVÁNÍ

Děkuji všem, kteří mi byli při studiu i psaní této práce k dispozici. Zejména pak jejímu vedoucímu Ing. Ivu Malému, Ph.D. za vstřícnost a dobré směrování.

PROHLÁŠENÍ

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

V Praze dne 24. května 2019

ABSTRAKT

Jízdní řády s odhadem zpoždění pomocí zpětné vazby uživatelů

Tato práce pojednává o konceptu mobilní aplikace zaměřené na sběr informací o zpoždění a mimořádnostech v provozu autobusů veřejné dopravy na základě toho, jak ji uživatelé používají, a jejím následném vývoji na platformě Xamarin.Android. Mobilní aplikace si vyměňuje informace s aplikací cloudovou na platformě ASP.NET Core, která je také součástí této práce. Celé řešení je zaměřeno zejména na použitelnost a uživatelskou přívětivost, proto je také v průběhu opakovaně uživatelsky testováno.

Klíčová slova: mobilní aplikace, veřejná doprava, autobus, zpoždění, mimořádnost, Android, .NET

ABSTRACT

Timetables with crowdsourced data about delays

This thesis describes the concept of a mobile application for crowdsource data collection of delays and disruptions in public bus transport, and its following development on the Xamarin.Android platform. The application exchanges data with a cloud based ASP.NET Core service, whose implementation is also a part of this thesis. This work focuses heavily on the application's user friendliness and usability and has therefore been tested with potential users in multiple stages of its development.

Keywords: mobile app, public transport, bus, delay, disruption, Android, .NET

OBSAH

1 ÚVOD

1.1 Motivace

Oblast veřejné hromadné dopravy mi byla vždycky blízká. Již od dětství to bylo spolu s informačními technologiemi něco, co mě svým způsobem fascinovalo i naplňovalo. Proto jsem se rozhodl v těchto oblastech rozvíjet své dovednosti, a to studiem i prací.

Ve veřejné hromadné dopravě vidím velký potenciál, který se týká dalšího rozvoje informačních technologií a jejich využívání. Tento potenciál existuje například v problematice plánování spojů, obratů vozidel i zaměstnanců dopravců, zejména v návaznosti na data o cestujících – jejich počtu, trasách jízdy, frekvenci cestování aj. – nebo na data z jiných informačních systémů. Já se ovšem ještě více zajímám o tvorbu softwaru, který využívá přímo veřejnost, v tomto případě cestující veřejnost. Konkrétně u dopravy, kde z podstaty věci jde o mobilitu, mají pro cestující největší smysl mobilní aplikace.

Mezi nimi již existuje mnoho různých řešení, která pomáhají s vyhledáváním spojení a zobrazují nejrůznější informace o jízdě. Nicméně ve většině případů jde technologicky pouze o výpis dat z databáze. Těch technologicky zajímavějších řešení je poskrovnu a zdaleka neexistují pro všechny druhy dopravy. Proto i zde vidím potenciál k dalšímu rozvoji technologií.

Na tomto základě jsem si jako základní téma své práce zvolil mobilní aplikaci pro cestující ve veřejné hromadné dopravě.

1.1.1 Oblast zaměření

Veřejnou hromadnou dopravu můžeme rozdělit na městskou a meziměstskou, která se dále dělí podle dopravních prostředků na železniční, silniční (autobusovou), leteckou a vodní. S tímto dělením pracuje také Český statistický úřad, jehož data jsem využil při volbě zaměření aplikace.

Druh dopravy	železniční	silniční	letecká	vodní	městská
Počet cestujících v tisících	183 163	329 316	6 657	813	2 317 315

Tabulka 1 – Počet cestujících ve veřejné hromadné dopravě v ČR za rok 2017 (1)

Na základě těchto dat vidíme, že zdaleka nejvíce cestujících využívá přepravu městskou dopravou, následovanou tou autobusovou. Vodní doprava je v České republice zastoupena minimálně, podobně i letecká přepravuje výrazně méně cestujících než ostatní druhy. Oba tyto jsou navíc velmi specifické.

Mým o něco důležitějším oborem zájmu v oblasti dopravy je doprava železniční, které se věnuji i ve svém volném čase, nicméně pro železniční dopravu je k dispozici několik aplikací, které jsou použitelné ze své podstaty v celé zemi. Jedním příkladem za všechny je aplikace

Můj vlak od Českých drah, které jsou národním dopravcem a mají v Česku drtivou většinu v počtu najetých kilometrů. (2) Tuto aplikaci si pouze na Google Play nainstalovalo více než 500 000 uživatelů a získala různá ocenění. Dále bych mohl zmínit aplikaci NaVlak, která není závislá na dopravci a nabízí informace z dat Správy železniční dopravní cesty relevantní pro celou republiku.

Naproti tomu u autobusové, a ještě spíše u městské dopravy existuje více dopravců a jejich výkony jsou více rozmělněné, i proto zde neexistují výrazněji vyčnívající aplikace, které by byly univerzálním řešením pro cestování kdekoli po Česku. Nesmíme zapomenout na aplikaci IDOS a jí podobné, které jsou ale relevantní pro všechny druhy dopravy, takže jsem je při výběru konkrétního zaměření nebral v potaz a budu se jim věnovat v části 2.1.

To jsou tedy mé hlavní důvody pro to zaměřit se na tvorbu aplikace pro využití v autobusové a městské dopravě. Přestože jisté zkušenosti z této oblasti mám, provedl jsem v souladu se zadáním předcházejícího samostatného projektu menší průzkum mezi cestujícími sloužící k inspiraci, na jaký konkrétní problém se zaměřit.

1.2 Požadavky cestujících

1.2.1 Volba cílové skupiny

Navzdory tomu, že mým tématem je aplikace pro široké spektrum uživatelů, jsem průzkum provedl v úzkém okruhu cestujících, ke kterému mám nejblíže, abych lépe porozuměl jejich požadavkům.

Využil jsem formu online dotazníku, který jsem distribuoval mezi uživatele sociálních sítí ve skupinách sdružujících obyvatele a příznivce města Mariánské Lázně. Je to středně velké město s 15 000 obyvateli, ze kterého pocházím.

Jeho síť městské hromadné dopravy není nikterak velká, zato ale vykazuje prvky sítí větších měst. Cestující mají na zastávkách k dispozici standardní zastávkové jízdní řády (v malých sítích jsou běžné méně přehledné linkové) a schéma linek, ve vozidlech jsou umístěny terminály pro nákup různých druhů jízdenek mj. bezkontaktní platební kartou, dopravní obslužnost zajišťuje dopravce vlastněný z většiny městem, který má zároveň vlastní webové stránky se základními informacemi o dopravě a přepravě. Městská doprava v Mariánských Lázních zároveň nemá žádnou vlastní mobilní aplikaci.

1.2.2 Cíl průzkumu

V otázkách průzkumu jsem se zaměřil na spokojenost se stávajícími aplikačními řešeními. Respondenty jsem žádal o posouzení následujících funkcí či vlastností, které by potenciální aplikace mohla mít, ve smyslu potřebnosti a užitečnosti:

- Informace o jízdném
- Informace o způsobu odbavení (nástup, návod k používání terminálu)

- Offline jízdní řády
- Jízdní řád pro celou linku na jedné stránce
- Zaručené informace, jak se dopravit k významným bodům ve městě (památky, prameny, úřady, obchody aj.)
- Plánky přestupních uzlů (Chebská, Úšovická, Nádraží)
- Upomínka, že se blíží zastávka, kde mám vystoupit
- Informace o výlukách a mimořádnostech
- Informace o místech pro předprodej jízdenek (otevírací doba, adresa, možnosti platby)
- Jedna aplikace se všemi potřebnými informacemi

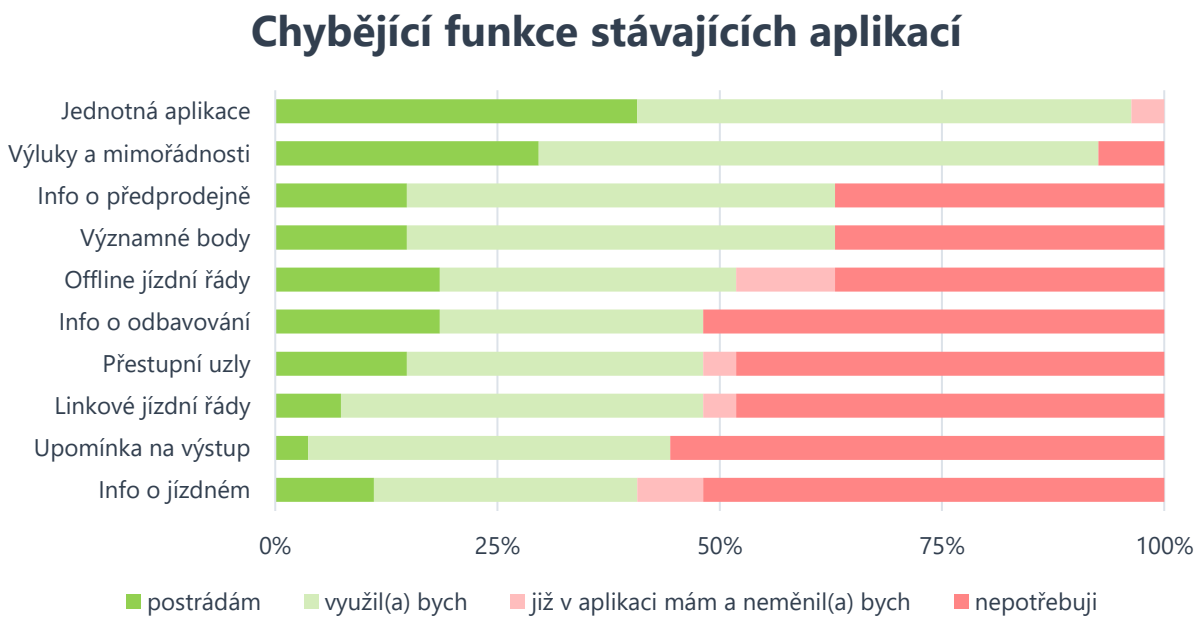
U každé z těchto položek mohli vybrat možnost *nepotřebuji*, *již v aplikaci mám a neměnil(a) bych*, *využil(a) bych*, nebo *postrádám*.

V průzkumu samozřejmě nechyběla závěrečná otevřená otázka *„Chcete něco sdělit k myšlence mobilní aplikace pro městskou dopravu?“*.

1.2.3 Vyhodnocení odpovědí

Do průzkumu se zapojilo celkem 39 respondentů. Z toho 87 % tvoří občané a stálí obyvatelé Mariánských Lázní. Je to dáno pochopitelně tím, kde byl průzkum distribuován. 92 % jsou majitelé chytrého telefonu, z nichž 94 % má internet v mobilu (87 % z celku). 23 % odpovídajících se označilo za nadšence v oblasti veřejné dopravy.

Pozitivní pro účely průzkumu je, že 90 % lidí uvedlo, že městskou dopravu v Mariánských Lázních využívá minimálně jednou za měsíc. Těch, kteří ji využívají téměř denně je pak 38 %.



Graf 1 – Chybějící funkce stávajících aplikací

Z posouzení funkcí a vlastností, které stávajícím aplikacím mohou chybět, vyplynulo, že respondenty nejvíce trápí absence jednotné aplikace obsahující veškeré potřebné funkce. Podíváme-li se na položku na pomyslné druhé příčce, zvláště ve světle situace v Mariánských Lázních a jiných podobných městech, vidíme, že jedna pro uživatele důležitá funkce ve všech zmíněných aplikacích chybí. Jde o informace o výlukách a mimořádnostech, které je můžou na cestě potkat.

Naopak nejmenší poptávka je po informacích o jízdném, odbavování, přestupních uzlech nebo upomínce na výstup. To je poměrně logické vzhledem k tomu, co již bylo řečeno, že většina respondentů jsou stálými obyvateli města a síť městské dopravy v něm patří mezi ty menší.

Ze závěrečné otázky vyplývá ještě nejméně jedna zajímavá informace. Uživatel poptává dochvilnost řidičů, jiný zas doplňuje, že by v aplikaci *neměla chybět online poloha vozidel, nebo alespoň aktuální zpoždění spoje*. Dovolil bych si tvrdit, že tento požadavek bude sdílet i velký počet z těch, kteří označili jako podstatnou funkci informace o mimořádnostech.

Zmíním ještě jeden delší příspěvek, kde se respondent nejprve táže *„Proč dělat aplikaci pouze pro MHD v Mariánských Lázních, která je pro jiná města nevyužitelná?“*. Žádá v něm vytvoření takového softwaru, který bude obsahovat offline jízdní řády pro celou Českou republiku.

1.3 Téma práce

Požadavek na aktuální informace z dopravy se umístil s 93 % žádanosti na druhém místě mezi funkcemi, které si přejí uživatelé stávajících aplikací, za požadavkem jednotné aplikace se všemi potřebnými funkcemi.

Ze své zkušenosti mohu potvrdit, že tyto informace nejsou v případě autobusové dopravy zdaleka samozřejmostí. V mariánskolázeňské městské dopravě se ovšem tolik mimořádností nekoná, proto předpokládám, že zde cestující promítli tento nedostatek zejména z meziměstské dopravy.

Navíc protože vývoj jednotné aplikaci považuji za překračující možnosti bakalářské práce, jsem se rozhodl zaměřit na vývoj aplikace řešící problematiku zpoždění a mimořádností v celé autobusové dopravě. K řešení lze přistoupit prakticky dvěma základními možnosti:

1. Agregací dat, která jsou již nyní sbírána jednotlivými dopravci a organizátory dopravy
2. Sběrem dat vlastních na základě využití mobilní aplikace

První přístup má nespornou výhodu v přesnosti a spolehlivosti dat. Druhý přístup má obrovskou výhodu ve své univerzálnosti a nezávislosti na zdrojích dopravců. Nevýhodou může být nepřesnost, a hlavně nutnost instalace aplikace do telefonu pro získání dat. Vzhledem k tomu, že jakmile dopravce zařízení pro sledování polohy má a má zájem jej zveřejnit, zapojuje se obvykle do existující sítě dat o zpoždění a aplikace typu IDOS tato data

následně zobrazují, nevidím příliš prostoru pro vytvoření nového aplikačního řešení na základě přístupu číslo 1.

Naopak druhým přístupem lze přinést něco, co v oblasti veřejné dopravy v ČR patrně ještě neexistuje.[1] Jde o moderní způsob práce s daty využívající skutečnost, že vlastnictví mobilního telefonu s připojením k internetu je v dnešní době obvyklé.[2] Tento způsob se anglicky nazývá *crowdsourcing* a v oblasti dopravy je běžně využíván poskytovateli mapových služeb ke zjišťování hustoty provozu na silnicích a nahlašování mimořádných událostí, tedy k funkcím, které chybějí v aplikacích pro veřejnou dopravu respondentům mého průzkumu.

Z těchto důvodů volím způsob číslo 2 a vytvořím aplikaci odhadující zpoždění na základě sesbíraných dat o polohách uživatelů a umožňující sdílet mimořádnosti v provozu.

[1] V rámci analýzy se to pokusím vyvrátit.
[2] 87 % respondentů mého průzkumu odpovědělo, že vlastní chytrý telefon s připojením k internetu.

2 ANALÝZA

2.1 Stávající mobilní aplikace

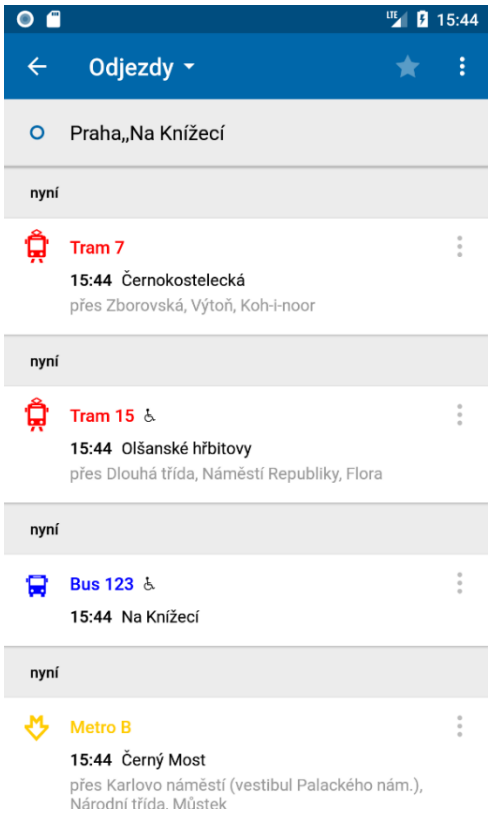
V současné době existuje několik mobilních aplikací, které lze využít i v městské dopravě. Jelikož Android je nejrozšířenější mobilní platforma (podle statistiky globální společnosti StatCounter drží k dubnu 2019 v České republice podíl 78,6 % trhu (3)), je cílovou platformou mé aplikace a zároveň hlavní obchod s aplikacemi této platformy (Google Play) jako jeden z mála zveřejňuje alespoň orientační údaje o počtu stažení aplikací, rozhodl jsem se při analýze stávajících aplikací zaměřit zejména na tuto platformu. Pro druhou nejrozšířenější platformu, iOS, s podílem 19,91 % v současné době neexistuje žádné významnější odlišné řešení, naopak některá z platformy Android zde chybí. Zmínit mohu ještě třetího zástupce, mobilní platformu Windows, jejíž podíl za poslední léta upadl k 0,76 %, jelikož společnost Microsoft, jakožto její autor, přestala vyvíjet nové funkce i nová zařízení a oznámila ukončení podpory 10. prosince 2019. Ostatní platformy mají ještě nižší tržní podíl.

Jak jsem již předeslal v úvodní části, nejvíce aplikací pro veřejnou dopravu je v kategorii obecných, které slouží uživatelům po celé republice. Těch specializovaných například na městskou dopravu není mnoho a obvykle nepřinášejí téměř žádnou přidanou hodnotu oproti aplikacím obecným. Proto s analýzou začnu právě v obecném segmentu.

2.1.1 Aplikace IDOS

Vývojář: CHAPS, spol. s r. o.
Vydavatel: MAFRA, a. s.
Počet stažení na Google Play: 1-5 mil.
Dostupnost: Android, iOS

Nejznámějším a nejrozšířenějším zástupcem mezi mobilními aplikacemi ve veřejné dopravě vůbec je aplikace IDOS. (4) Vývoj zajišťuje společnost CHAPS, která je dlouhodobě správcem údajů o jízdních řádech v České republice.



Obrázek 1 – Aplikace IDOS, verze 2.5.1

FUNKCIONALITA

IDOS umožňuje vyhledávat spojení mezi všemi dopravními prostředky, stejně tak lze při vyhledávání prostředky filtrovat, nebo vyhledávat v různých souborech jízdních řádů, mezi které patří Všechny jízdní řády, Vlaky + Autobusy, Vlaky, Autobusy a jednotlivé integrované dopravní systémy a sítě MHD. Kromě toho lze vyhledávat spojení podle adresy, některých bodů zájmu a výběrem zastávky na mapě.

U spojů lze zobrazit jejich detail, pokud dopravce zadal informace o svém tarifu do státní databáze, stejně tak, jestliže je dopravce zapojen do centrálního systému pro sdílení údajů o zpoždění a poloze vozů, jsou i tyto údaje v aplikaci zobrazeny.

Mimo vyhledávání spojení najdeme v aplikaci vyhledávání odjezdů z určité stanice, (Obrázek 1) v minulých letech přibyla funkce pro přímý nákup jízdních dokladů pro vybrané meziměstské autobusy zapojené do prodejního systému AMSBus, který dnes již taktéž skrze svojí dceřinou společnost provozuje firma CHAPS. Navíc je v aplikaci k dispozici mnoho šablon pro nákup SMS jízdenek.

SHRNUTÍ

Aplikace poskytuje informace velkému množství uživatelů. Je postavena na základech osvědčené webové aplikace a obsahuje veškeré základní informace pro klasické naplánování spojení. Pokročilejší funkcionalita, jako jsou informace o zpoždění nebo nákup jízdenek, zde je, nicméně u omezeného množství spojů, protože je nutná součinnost dopravce.

Je dostupná zdarma s reklamami, které je možné vypnout za 49 Kč na rok nebo za 149 Kč trvale.

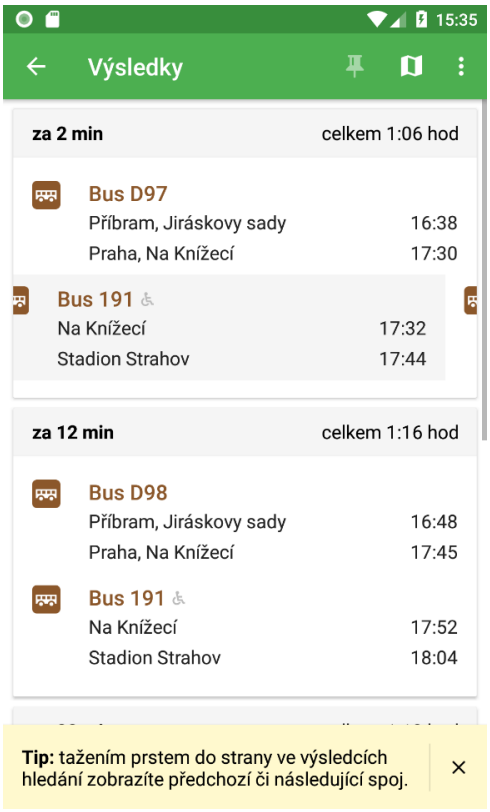
2.1.2 Aplikace Jízdní řády

Vydavatel: Seznam.cz, a. s.
Počet stažení na Google Play: 500 tis.-1 mil.
Dostupnost: Android, iOS

Druhou příčku v pořadí podle počtu stažení obsadila aplikace Jízdní řády od Seznamu. (5) Ten ji nevyvinul od začátku, ale postavil na aplikaci zvané Pubtran, kterou koupil v roce 2015. Pubtran, podobně jako mnoho dalších, měl problémy s přístupností dat o jízdních řádech, kterým se budu věnovat v další části. (6; 7)

FUNKCIONALITA

Rozhraní této aplikace je o něco jednodušší a dle mého názoru i přehlednější než v případě IDOSu. Aplikace Jízdní řády jako první přišla s možností už ve výsledcích vyhledávání swipe gestem přepínat jednotlivé spoje na pozdější či dřívější (Obrázek 2). Tuto funkci v první polovině roku 2019 převzal i IDOS nebo Můj vlak. Z obrazovky s detailem spojení se lze prokliknout například na možnost nákupu SMS jízdenky.



Obrázek 2 – Aplikace Jízdní řády, verze 5.12.7

V některých případech se zobrazují informace o zpoždění spoje, nicméně počet těchto případů je objektivně ještě nižší než u aplikace IDOS. Můj názor je, že je to dáno individuálním vyjednáváním vydavatele s poskytovateli dat a neveřejnou dostupností zdrojů, které využívá IDOS, ale nepodařilo se mi k tomuto dohledat žádné zdroje.

V Jízdních řádech nechybí ani možnost filtrování podle dopravních prostředků (nelze ale vyhledávat pouze v konkrétní síti MHD), vytváření oblíbených položek nebo přehledná historie vyhledaných spojení. Nenalezneme zde ale tabule s odjezdy z jednotlivých zastávek ani možnost přímého nákupu jízdenek.

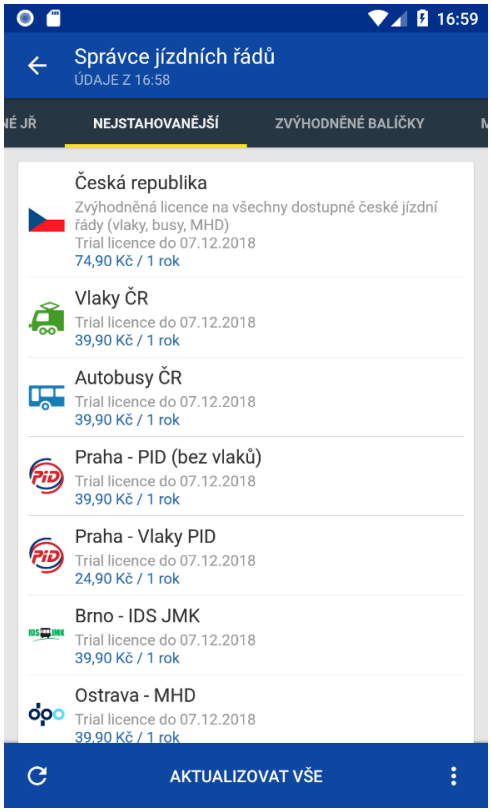
SHRNUTÍ

Tato aplikace zaujme svým lehkým vzhledem. Na něj i na své další funkce nalákala taktéž velký počet uživatelů, i přesto, že osobně jsem žádnou velkou reklamní kampaň na tuto aplikaci nezaznamenal. Předpokládám, že to z velké části bude tím, že se na Google Play jmenuje zkrátka „Jízdní řády", a tak se objevuje po vyhledání tohoto výrazu na prvním místě. Uživatelské hodnocení v této službě má velmi podobné jako konkurence od firmy CHAPS. Dalších funkcí najdeme v aplikaci málo, proto ji považuji za velmi podobnou právě IDOSu.

Jízdní řády od Seznamu jsou k dispozici zdarma a bez reklam.

2.1.3 Aplikace CG Transit

Vývojář: circlegate
Vydavatel: circlegate
Počet stažení na Google Play: 100-500 tis.
Dostupnost: Android, iOS, Windows



Obrázek 3 – Aplikace CG Transit, verze 3.1.4

CG Transit vyvinuli dva absolventi naší fakulty – Jan Blažek a Zbyněk Říha. (8) Jejím hlavním cílem je nabídnout jízdní řády dostupné bez připojení k internetu.

FUNKCIONALITA

V aplikaci opět nechybí žádné základní funkce. Kromě vyhledávání spojení s nejrůznějšími parametry nabízí možnost zobrazení odjezdů ze zastávek, výběr zastávky na mapě, a navíc i seznam linek dané sítě MHD.

Oproti předchozím aplikacím tedy opravdu zaujme podporou offline jízdních řádů. Ty lze stahovat po různých balíčcích (Obrázek 3) dělených na městské sítě MHD, meziměstské autobusy ČR, autobusy SR, vlaky v mnoha evropských zemích dělené podle

států či skupin států, všechny evropské vlaky nebo kompletní balíčky všech druhů dopravy na území Česka, Slovenska.

SHRNUTÍ

Tato aplikace jednoznačně vítězí v oblasti jízdních řádů dostupných offline. Dělení podle měst umožňuje příjemnější využití v MHD. Nevýhodou je, že pravděpodobně také kvůli špatné dostupnosti dat o jízdních řádech v Česku je nutné za aplikaci platit. Poplatky jsou roční a cena se odvíjí od požadovaných balíčků jízdních řádů. Balíček MHD v malém a středním městě stojí 24,90 Kč za rok, u velkého města to je 39,90 Kč, stejně tak u balíčku všech meziměstských autobusů nebo vlaků v Česku. Licence na balíček pro celý stát se všemi druhy dopravy stojí 74,90 Kč a kompletní data pro Česko, Slovensko a evropské vlaky mají cenu 124,90 Kč za rok.

Po první instalaci má uživatel nárok na zkušební licenci zdarma na všechny balíčky platnou jeden měsíc a v aplikaci nejsou umístěny žádné reklamy.

2.1.4 Aplikace MHD Tabule

Vývojář: Petr Pyszko
Vydavatel: CHAPS, spol. s r. o.
Počet stažení na Google Play: 100-500 tis.
Dostupnost: Android

MHD Tabule jsou příkladem první aplikace určené přímo pro městskou dopravy. (9) Vyvinul ji nezávislý vývojář, avšak zřejmě z licenčních důvodů přešla pod společnost CHAPS.

FUNKCIONALITA

Zatímco předchozí aplikace byly funkčně velmi podobné, tato aplikace se soustředí na jedinou věc. Tou je zobrazování odjezdových tabulí ze zastávek, (Obrázek 4) případně zobrazování tabulí s uživatelsky předdefinovanými spojeními. Takovéto tabule lze zobrazovat jako widget ve spouštěči aplikací.

Obrázek 4 – Aplikace MHD Tabule, verze 1.4.3

Výhodou však je možnost stáhnout si jízdní řád konkrétních tabulí pro použití offline. Stažená data ale mají omezenou platnost (obvykle 2 týdny) a do té doby je nutné je znovu aktualizovat, jinak je není možné dále používat.

SHRNUTÍ

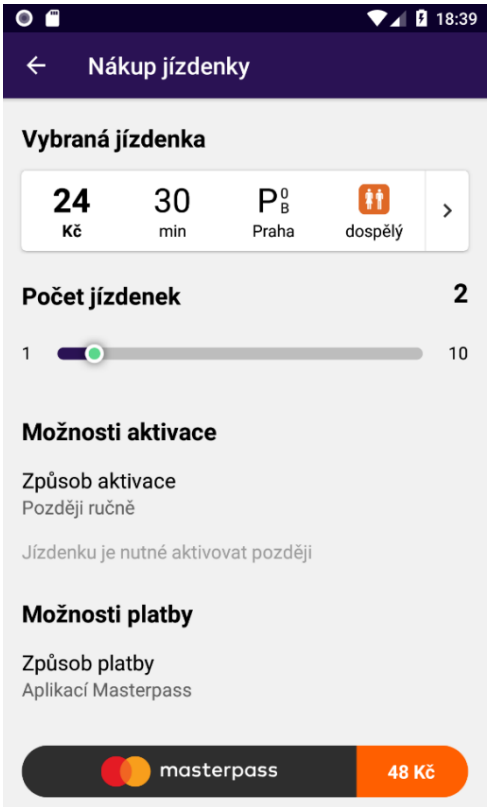
Tato aplikace neoplývá funkcemi. K dispozici nejsou žádné online informace o spojích a plynulosti provozu, nicméně i přesto zaujme, jelikož lze po stažení tabulí používat bez připojení k internetu, takže i tak může směle konkurovat placenému CG Transitu. Bohužel aplikace nebyla aktualizována od 16. května 2016, přesto je naštěstí plně funkční.

MHD Tabule jsou dostupné zdarma a bez reklam.

2.1.5 Aplikace PID Lítačka

Vývojář: Operátor ICT, a. s.
Vydavatel: Hlavní město Praha
Počet stažení na Google Play: 100-500 tis.
Dostupnost: Android, iOS

Hlavním zástupcem softwaru vyvinutého pro potřeby konkrétní městské sítě je PID Lítačka sloužící cestujícím v Pražské integrované dopravě. (10) Jde o poměrně mladý přírůstek vyvinutý z původní aplikace DPP Info později přejmenované na PID Info.



Obrázek 5 – Aplikace PID Lítačka, verze 3.2.0

FUNKCIONALITA

Po funkční stránce můžeme PID Lítačku snadno připodobnit k IDOSu omezenému na pražskou dopravu a přebarvenému do modro-zelena. Najdeme zde opět vyhledávač spojení, přehled odjezdů i mapu se zastávkami. Vše vzhledově dobře zapadá do systému PID a karty Lítačka. Na obrazovce s detailem spojení se uživatel dozví informace o mimořádnostech a u spojů, které nevypravuje Dopravní podnik Hlavního města Prahy (DPP), jsou k dispozici také informace o zpoždění.

Hlavním benefitem je možnost nákupu a správy elektronických jízdenek. (Obrázek 5) Tyto jízdenky mají podobu 2D kódu zobrazovaného uvnitř aplikace a je možné je při technických potížích ověřit náhradním způsobem pomocí tzv. pohyblivých obrazců. Jízdenka se aktivuje dle volby uživatele, buď ihned po nákupu, v předem vybraný čas, nebo na vyžádání. Ve všech případech se uplatňuje ochranná lhůta min. 2 minuty od okamžiku nastavení aktivace.

Kromě toho lze v aplikaci zobrazit různé plánky a mapky, seznam bezbariérových stanic včetně omezení jejich provozu, další omezení v dopravě a výluky, které ale na mě působí způsobem, že pouze stahují RSS zdroj a otevírají informace v nějakém webovém okénku, takže tato funkce graficky příliš nezapadá do vzhledu aplikace.

SHRNUTÍ

Aplikace PID Lítačka je dobrým řešením pro cestující v Pražské integrované dopravě. Poskytuje nejen všechny základní informace, ale i další o zpoždění a mimořádnostech. Nicméně zpoždění není k dispozici u jádra celého systému – pražské MHD.

PID Lítačka je taktéž zdarma a bez reklam.

2.2 Závěry analýzy

Stávající mobilní aplikace můžeme rozdělit na obecné, které cílí na uživatele všech druhů veřejné dopravy, a ty, které jsou určeny přímo pro MHD. Funkčně se však liší jen malá část z nich. Všechny umí nějakým způsobem vyhledávat spoje. Většina využívá data ze stejného zdroje, který je ale mnohdy omezený (v případě zpoždění a mimořádností). Více se odlišuje svým konceptem aplikace MHD Tabule, která poskytuje informace offline, ale nemá mnoho funkcí a pravděpodobně není dále vyvíjena. Pro klasickou dopravní aplikaci s offline daty můžeme sáhnout po CG Transitu, ten je ale zase zpoplatněný. Přímo pro MHD navíc existuje více aplikací specifických pro různá města s podporou dalších funkcí, jako je nákup jízdenek. V této kategorii existuje například PID Lítačka.

Z analýzy stávajících řešení dále vyplynulo, že jedním z nedostatků, který se objevuje u všech analyzovaných aplikací, jsou informace o zpoždění a mimořádnostech. V tomto ohledu nejlépe obstála aplikace PID Lítačka, která se ale soustředí pouze na jedinou oblast. I přesto v současné době stále nejsou dostupné informace o zpoždění v jejím jádru, tedy MHD Praha. Ostatní aplikace žádné informace o mimořádnostech v autobusové dopravě nezobrazují. V některých městech informace o zpoždění nalezneme, ale není to zdaleka pravidlem. Naopak žádná analyzovaná aplikace nesbírá data metodou crowdsourcingu.

Pro mé řešení to znamená, že se potvrdila má domněnka z úvodní části, že tato práce nemá mezi aplikacemi využívanými v Česku odpovídající konkurenci. To je pozitivní zpráva pro její smysluplnost, avšak znamená to, že je třeba dbát zvýšenou opatrnost na její použitelnost.

3 NÁVRH

Základním prvkem mé práce bude mobilní aplikace zprostředkovávající veškerou interakci mezi uživatelem a systémem. Ta uživateli umožní vyhledat zastávky, zobrazit odjezdy autobusů z těchto zastávek a vyhledávat jednoduchá spojení. Jestliže si člověk nějaký autobus vybere, může v aplikaci označit, že do něj nastoupil. Tím začne být sledována jeho poloha spárovaná s daným konkrétním spojem.

Nejpozději v tuto chvíli přichází na řadu nutnost přenosu dat přes internet. Dosáhnout požadovaného výsledku je možné prakticky dvěma způsoby; buďto komunikačním modelem klient-klient, který by ale kladl zbytečné nároky na internetový provoz a také výpočetní výkon uživatelských zařízení, nebo modelem klient-server, jehož základem je cloudová aplikace sloužící jako backend spravující data o jízdních řádech, zpoždění a mimořádnostech. Věřím, že je zřejmé, že v tomto ohledu nemá smysl pouštět se do nějakých experimentů, a že je tedy pochopitelné, že sáhnu po druhé možnosti, kterou využívají i všechny analyzované aplikace.

3.1 Mobilní aplikace

Za účelem co nejsrozumitelnějšího uživatelského rozhraní vytvořím aplikaci v souladu s nejnovějším oficiálním designem platformy Android známým pod názvem Material. (11) Pro návrh uživatelského rozhraní jsem zvolil nástroj Balsamiq Mockups v nejnovější verzi 3.5.16.

Navrhnul jsem 5 základních obrazovek shrnujících to nejdůležitější, co by uživatel měl mít vždy po ruce. Jde o obrazovky s vyhledáváním, naposledy zobrazenými zastávkami a autobusy, nejblíže umístěnými zastávkami a autobusy vzhledem k poloze uživatele, přehledem všech mimořádností, a hlavně aktuálním autobusem, ve kterém uživatel cestuje. V následující části práce je spolu se čtyřmi dalšími podrobněji představím.

Vzhledem k tomu, že obrazovek není mnoho, jsem ze tří základních možností pro navigaci mezi obrazovkami stejné úrovně (Lateral navigation) zvolil dolní navigační panel. Díky tomu budou všechny základní obrazovky rychle dostupné a uživatel nebude muset při navigaci otevírat žádnou další nabídku.

3.1.1 Obrazovka Vyhledávání

Obrazovka s vyhledáváním bude rozvržena do čtyř částí. Nejvýše bude nastavitelné datum a čas vyhledávání, níže možnost otevřít přehled konkrétní linky či spoj, jestliže si uživatel pamatuje jeho číslo. V třetí části se bude nacházet vyhledávání odjezdů ze zadané zastávky a poslední možností bude vyhledávání přímých spojů mezi dvěma zastávkami.

Možnost vyhledávání přestupních spojení by byla samozřejmě vhodnou volbou pro tuto aplikaci, avšak v současné době neexistuje bezplatné API některé služby třetí strany

(např. IDOS), které bych k tomu využil, a po technické stránce bych se v této práci raději věnoval inovativnějším funkcím nežli vývoji dalšího algoritmu pro hledání spojení.

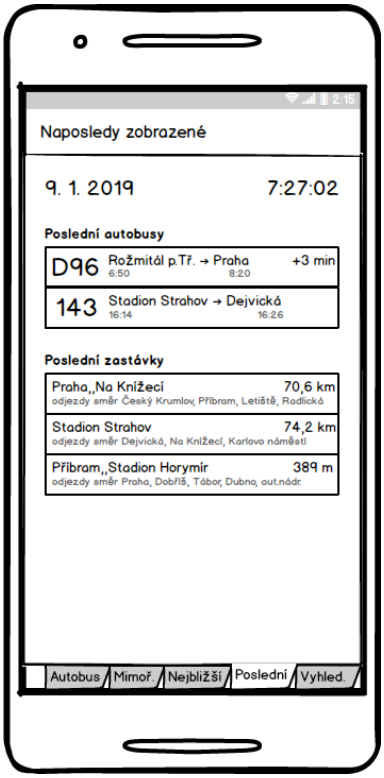
3.1.2 Obrazovka Poslední

Na této obrazovce se bude nacházet přehled autobusů a zastávek, které uživatel naposledy otevřel. U autobusů bude zobrazeno označení linky, výchozí a konečná stanice s časy výjezdu a příjezdu a aktuálním zpožděním, je-li dostupné. V případě posledních zastávek je k dispozici informace o tom, do jakých směrů z nich lze cestovat a jak vzdálené jsou od aktuální polohy zařízení. V horní části je viditelně umístěno aktuální datum a čas včetně vteřin, jelikož řidiči mají čím dál tím častěji k dispozici přesný čas a musí se podle něj řídit.

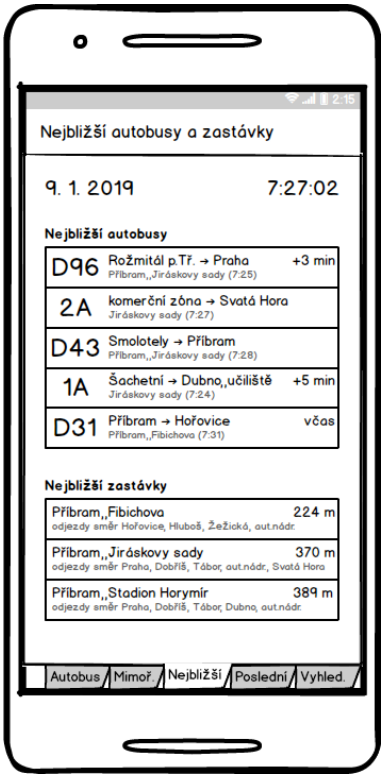
3.1.3 Obrazovka Nejbližší

Návrh obrazovky Nejbližší je velmi podobný jako v předchozím případě 3.1.2. Nachází se na něm přehled autobusů, které jsou v dané chvíli uživateli nejblíže s tím rozdílem, že místo informace o výjezdu a příjezdu z výchozí a do cílové zastávky je na druhém řádku informace o nejbližší zastávce, ve které je možnost do autobusu nastoupit s časem pravidelného odjezdu. Ve spodní části je přehled nejblíže umístěných zastávek ve zcela shodném formátu. Stejně tak je na stránce zobrazeno aktuální datum a čas. Při změně polohy se zastávky a autobusy automaticky aktualizují, navíc zažitým gestem potažení prstem shora dolů je možné aktualizaci vyvolat ručně.

Obrázek 6 – Prototyp Vyhledávání



Obrázek 7 – Prototyp Poslední



Obrázek 8 – Prototyp Nejbližší

3.1.4 Obrazovka Mimořádnosti

Zde se zobrazí všechny mimořádnosti, se kterými bylo interagováno za posledních 24 hodin. Toto zobrazení bude opět možné stejným gestem ručně aktualizovat. Rozložení jednotlivých mimořádností vychází z jejich detailního zobrazení (3.1.8) a skládá se ze čtyř základních svisle rozmístěných prvků: záhlaví s ikonou, druhem události a vyjádřením celkového hodnocení od uživatelů, dále název lokality, slovní popis události vložený uživatelem a zkrácený na maximálně 3 řádky a čtyři ukazatele času poslední interakce s událostí – zleva nahlášení, kladné ohodnocení, záporné ohodnocení a okomentování. Na displeji vpravo dole bude umístěno plovoucí tlačítko sloužící k nahlašování nových událostí.

3.1.5 Obrazovka Můj autobus

Poslední z těchto pěti hlavních návrhů obsahuje detail autobusu, ve kterém uživatel bude v daný moment cestovat. Jestliže se v žádném nebude nacházet, tak se na této stránce zobrazí hláška, která jej na to upozorní a navede, jakým způsobem si autobus vyhledat.

V horní části se nachází označení linky, úplné (licenční) číslo a číslo spoje, vpravo pak jeho cílová zastávka. Pod tyto informace jsem navrhl umístit stručný přehled jízdního řádu obsahující výchozí stanici, cílovou stanici, název místa, kde se autobus nachází, a dvě nejbližší zastávky vzhledem k aktuální pozici vozu. Na každém řádku tohoto přehledu je vpravo údaj o pravidelném odjezdu z daného místa – v případě aktuální pozice jiné než v zastávce se jedná o údaj odhadnutý – s číslem vyjadřujícím počet minut zpoždění spoje, je-li pro dané místo k dispozici. Klepnutím na tento seznam je možné jej rozbalit do kompletního jízdního řádu, kde se údaj o zpoždění nezobrazuje v minutách, ale jako čas odjezdu z daného místa (obvykle zastávky).

V dolní polovině displeje se znovu nachází informace o zvolené výstupní zastávce uživatele s detaily aktuálního zpoždění. Na ni je možné klepnout a volbu změnit. Pod touto informací je umístěn seznam případných mimořádností, které se týkají daného autobusu, v jejichž záhlaví je tlačítko pro nahlášení nové.

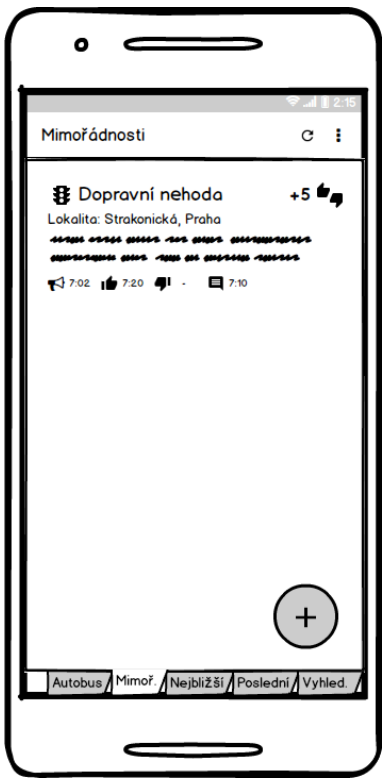
Nerad bych opomenul panel nástrojů v záhlaví obsahující tlačítko s ikonou autobusu pro změnu výstupní zastávky a předčasné ukončení jízdy, tlačítko pro otevření mapy s vyznačenou trasou spoje a tlačítko pro aktualizaci obrazovky se všemi údaji na ní zobrazenými.

3.1.6 Obrazovka Přehled jízdy autobusu

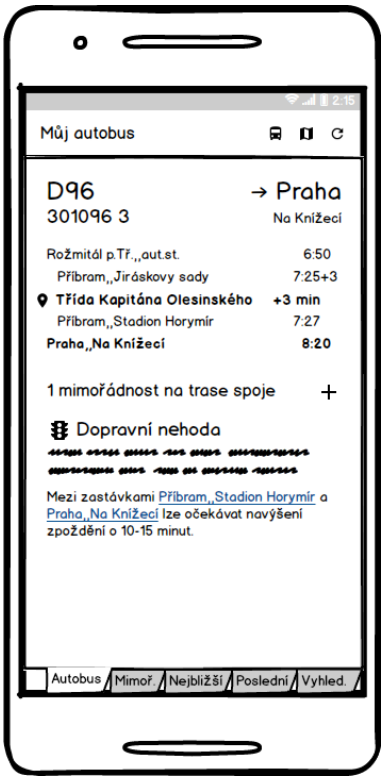
Tato obrazovka poskytuje detailní informace o jízdě autobusu uživateli, který se na ni může dostat například přes vyhledávání, poslední nebo nejbližší spoje.

Rozložení této obrazovky je až na výjimky shodné s předchozí. Rozdíl je zejména v tom, že oproti Mému autobusu zde nemusí být vždy dostupná informace o zpoždění spoje. V takovém případě zkrácený přehled vybere zastávky pouze na základě odhadu aktuální

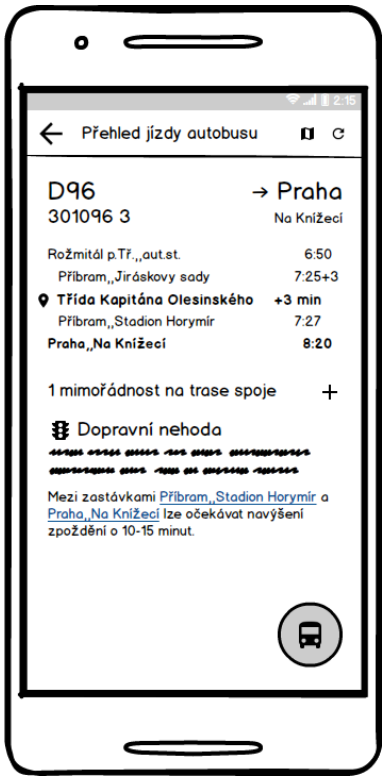
polohy autobusu podle jízdního řádu. V tom případě se taktéž nezobrazí dodatečná informace o aktuálním zpoždění. Na této obrazovce taktéž pozbývá smyslu informace o výstupní zastávce, proto zde není uvedena. Naopak navíc je v pravém dolním rohu umístěno plovoucí tlačítko s ikonou autobusu, které uživatel stiskne, jestliže do autobusu nastoupí. Tím se spoj nastaví jako Můj autobus a aplikace začne sledovat jeho polohu. Jestliže to nebude z kontextu zřejmé, bude uživatel vyzván k výběru cílové zastávky, aby existovala v systému informace, kdy uživatele přestat sledovat.



Obrázek 9 – Prototyp Mimořádnosti



Obrázek 10 – Prototyp Můj autobus



Obrázek 11 – Prot. Přehled autobusu

3.1.7 Obrazovka Detail zastávky

V Detailu zastávky nalezneme mapku s její polohou a odjezdy autobusů. Na obrazovku povede cesta opět z vyhledávání nebo z posledních či nejbližších zastávek. Jednotlivé odjezdy budou zobrazeny jako číslo linky s konečnou stanicí a vybranými významnými nácestnými zastávkami.

3.1.8 Obrazovka Detail mimořádnosti

Obrazovka s detailem mimořádnosti bude přístupná po kliknutí ze stránky Přehled jízdy autobusu, Můj autobus nebo pochopitelně z obrazovky Mimořádnosti.

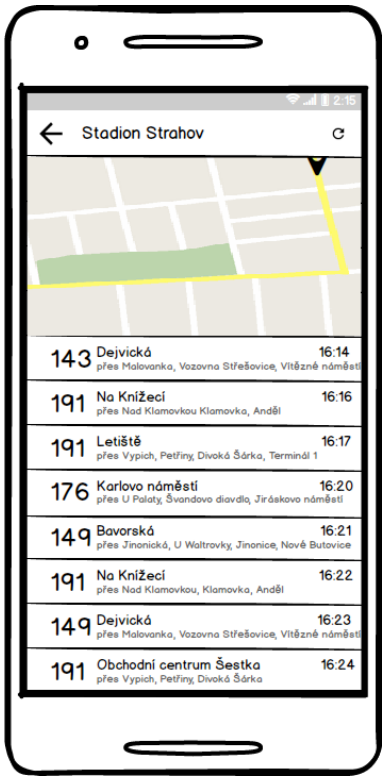
Návrh obsahuje mapku s místem události, nadpis, tlačítka pro hodnocení relevantnosti ostatními uživateli, o řádek níže textový popis místa události, pod kterým se nachází piktogramy informující o času posledních interakcí. Zleva: čas, kdy byla událost nahlášena,

čas posledního kladného hodnocení, čas posledního negativního hodnocení a čas posledního komentáře.

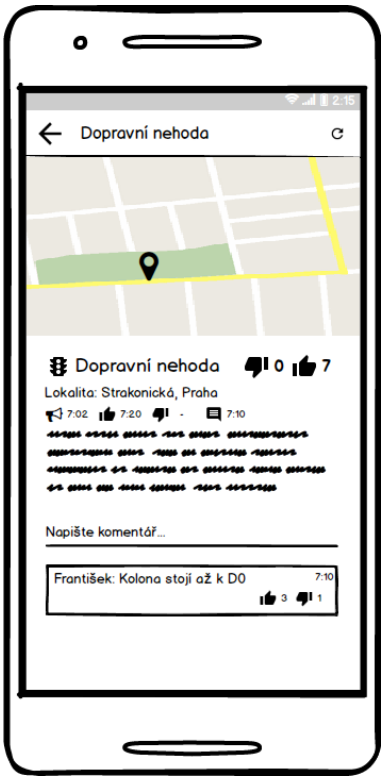
Komentáře je možné přidávat pod popisem události. Při prvním odesláním komentáře bude uživatel vyzván k zadání svého jména. Pod políčkem k přidávání komentářů je výpis všech těch již vložených. Komentáře je možné taktéž hodnotit.

3.1.9 Obrazovka Přidat mimořádnost

Při kliknutí na tlačítka sloužící k nahlášení nové mimořádnosti se v aplikaci otevře tato obrazovka. Bude obsahovat běžné ovládací prvky pro výběr kategorie, výši očekávaného zpoždění (kterou není nutné vyplnit) a slovní popis události. Místo, kde se mimořádnost stala, bude možné vybrat přímo na mapě nebo zadat slovně po kliknutí na název lokality. K mimořádnosti půjde přiřadit několik ovlivněných linek (je nutné vybrat ovlivněný směr) a bude možné také označit, že ovlivňuje i další linky ve stejné lokalitě.



Obrázek 12 – Prot. Detail zastávky



Obrázek 13 – Prot. Detail mimořád.

Obrázek 14 – Prot. Přidat mimořád.

3.2 Cloudová aplikace

Jak již z úvodu části Návrh vyplynulo, pro komfortní funkčnost celého systému je třeba vyvinout cloudovou aplikaci se třemi základními funkcionalitami, kterými jsou zpracování dat o jízdních řádech, uchovávání mimořádností a zpracování dat o zpoždění autobusů. V této části je také podrobněji popíšu.

3.2.1 Jízdní řády

Správa dat o jízdních řádech není hlavní funkcionalitou mé práce, tou je samozřejmě řešení zpoždění, avšak je velmi stěžejní pro fungování celého systému. Smyslem této funkcionality je uchovávat aktuální jízdní řády a poskytovat je klientským zařízením tak, aby se minimalizovaly nároky na jejich výpočetní výkon.

Jednou denně se z veřejně dostupného zdroje tato data stáhnou a zpracují tak, aby byla snadno dostupná pro všechny potřebné scénáře využití. Těmi jsou:

- Získání odjezdů ze zadané zastávky
- Získání přímých spojení mezi zadanými zastávkami
- Získání podrobností o vybraném spoji
- Získání zastávek nejblíže umístěných k zadané poloze
- Získání spojů nejblíže umístěných k zadané poloze
- Získání výsledků vyhledávání mezi zastávkami pro zadaný řetězec
- Získání výsledků vyhledávání mezi linkami pro zadaný řetězec

To znamená, že data se uloží v podobě, která zajistí, aby byla rychle dostupná například na základě času odjezdu a vybrané zastávce.

3.2.2 Mimořádnosti

Tento modul v porovnání s ostatními nebude obsahovat žádnou zásadní logiku pro zpracovávání dat. Základem je uchovávání jednotlivých mimořádností a jejich poskytování podle jednotlivých ovlivněných spojů. Tyto mimořádností musí být možné smazat nahlašujícím uživatelem a přidávat k nim kladná či záporná hodnocení a komentáře. Komentáře bude opět možné jejich autorem zpětně smazat.

3.2.3 Zpoždění

Jak jsem již předeslal, tato funkcionalita je v mých aplikacích tou hlavní. Cloudová aplikace bude zaznamenávat a uchovávat polohy uživatelů na základě jejich jednoznačného identifikátoru a zadaného autobusového spoje. Následně při zaslání požadavku na informace o zpoždění autobusu tyto zaznamenané polohy promítne na trasu spoje. Data vyhodnotí zvlášť pro každé zdrojové zařízení, odhadne pro ně zpoždění v zastávkách a ta nakonec zprůměruje. Tím se zajistí co nejvyšší přesnost výsledných informací.

Podrobněji se tomuto i ostatním modulům budu pochopitelně věnovat v části číslo 3 Implementace.

3.3 Testování prototypu

Pro testování návrhu uživatelského rozhraní mobilní aplikace jsem si vytvořil 5 testovacích scénářů v podobě částečně navazujících úkolů. Jejich cílem bylo zjistit srozumitelnost uživatelského rozhraní a to, jakým způsobem uživatel postupuje k dosažení cíle.

1. Otevřít autobus, který přijede ze Stadionu Strahov na Dejvickou kolem 16:30
2. Zaznamenat nástup do autobusu (nastavit spoj jako *Můj autobus*)
3. Nahlásit mimořádnost
4. Nastavit výstupní stanici
5. Ručně zaznamenat výstup z autobusu (odebrat spoj ze zobrazení *Můj autobus*)

V závěru jsem testera požádal o libovolný průchod zbývajícími částmi prototypu a komentář k nim.

Provedl jsem nejprve jeden zkušební test, který jsem nezaznamenával. Sloužil k ověření celistvosti prototypu, při něm jsem zjistil určité nedostatky a před hlavním testováním je opravil. V hlavní části jsem provedl celkem 4 testy, z nichž 1 dopadl neúspěšně. Testující uživatel byl zjevně nervózní, neřídil se mými pokyny a jednal zbrkle. Není z tohoto testu proto žádný smysluplný výstup.

Zde uvádím stručný přehled průběhu testování:

Úkol	Tester č. 1	Tester č. 2	Tester č. 3
1. **Otevření autobusu**	**Neúspěch** Nedaří se přepnout na příjezdy, zmatený ze stránky přímých spojů.	**Neúspěch** Otevírá odjezdy, nedaří se přepnout na příjezdy, licenční číslo linky a spoje ho vyrušuje.	**Částečný úspěch** Vyhledání bez problémů, není si však jistý svým umístěním ani splněním cíle.
2. **Záznam nástupu**	**Částečný úspěch** Nejistota, doporučuje změnit ikonu na „lokátor“.	**Částečný úspěch** Déle hledá, ikona by měla vyjadřovat přidání (např. plus).	**Úspěch**
3. **Nahlášení mimořádnosti**	**Úspěch** Navrhuje přidat automaticky ovlivněnou linku podle linky Mého autobusu.	**Úspěch** Není mu prvoplánově jasné, že na mapu lze umisťovat špendlík.	**Úspěch**

4.	**Nastavení výstupní stanice**	**Částečný úspěch** Není zřejmé, že na jízdní řád lze klepnout.	**Úspěch**	**Úspěch** Okamžitá jasná reakce.
5.	**Záznam výstupu**	**Úspěch** Popisek tlačítka však není srozumitelný.	**Neúspěch** Přemýšlí o správném tlačítku, ale nekliká na něj, nechápe koncept Mého autobusu.	**Neúspěch** Zdlouhavé hledání, po nápovědě nachází tlačítko, popisek není adekvátní.

Tabulka 2 – Stručný přehled chování testerů prototypu

V závěrečné fázi testu jsem se všech uživatelů ptal na stránku s detailem mimořádnosti. Ta byla všemi shledána srozumitelnou a povedenou. Jediný uživatel nepochopil ikonky s časy, po vysvětlení významu však uvedl, že ukazují užitečné informace. Dále jsem získal podněty k zařazení dostupných dat o mimořádnostech z veřejných zdrojů dopravců (např. v případě PID), k vizuálnímu oddělení tří částí obrazovky Vyhledávání a informování uživatele o skutečnosti, že časový údaj uvedený na některých stránkách není přesný, ale je použit ten ze zařízení.

3.3.1 Závěry testování

Z testování vyšlo najevo, že obrazovka Vyhledání potřebuje několik úprav. Za prvé to je zviditelnění možnosti přepínání mezi vyhledáváním odjezdů a příjezdů a za druhé si žádá lepší vizuální oddělení, protože obsahuje tři nezávislé vyhledávací formuláře. Ke zvážení je změna ikony na stránce Přehled jízdy autobusu (3.1.6) (Obrázek 11), aby lépe vyjadřovala svůj smysl. Stejně tak možnost předčasného vystoupení z autobusu vyžaduje změnu popisku a ke zvážení je lepší umístění. Naopak funkce nahlašování mimořádností je zcela srozumitelná a jádro funkce nastavení výstupní stanice taktéž.

Společným jmenovatelem u všech testerů byla lehká nejistota či drobné nejasnosti týkající se prvků uživatelského rozhraní. Řešením by mohlo být vytvoření *onboardingu* – tedy průvodce, který uživateli při prvním spuštění aplikace ukáže důležité ovládací prvky a jejich význam.

4 IMPLEMENTACE

4.1 Platformy

Vývoj aplikace pro operační systém Android probíhá typicky v prostřední Android Studio v jazyce Java či Kotlin. Existuje ale i možnost vývoje pomocí technologií .NET na platformě Xamarin. Ta v sobě navíc skýtá možnost sdílení kódu napříč platformami, takže můžete použít stejný datový model pro Android, iOS, Windows nebo macOS. (12) Toho by bylo možné snadno využít při dalším rozšiřování práce. Jelikož s platformou Xamarin již zkušenosti mám, zvolil jsem tuto možnost.

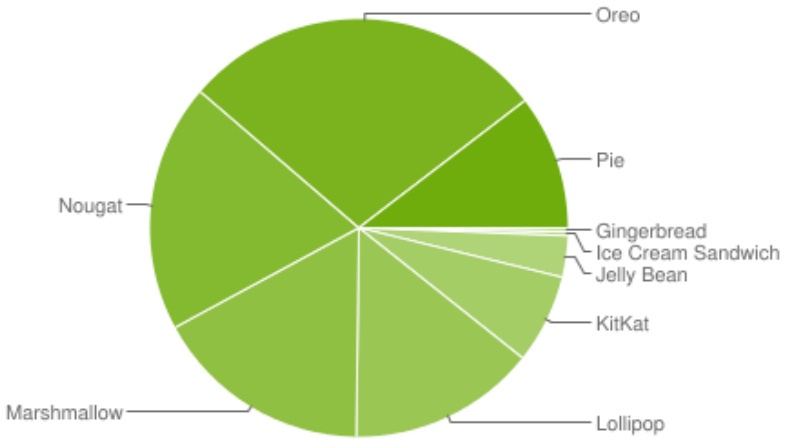
Xamarin nabízí dvě základní možnosti, jak mobilní aplikace pro Android vyvíjet:

1. Pomocí Xamarin.Forms se sdíleným uživatelským rozhraním s ostatními platformami
2. Pomocí Xamarin.Android s nativním uživatelským rozhraním

Jelikož jsem v části Návrh určil, že by aplikace měla dodržovat standardy nativního Material designu, zvolil jsem druhou možnost, díky které jsem uživatelské rozhraní aplikace mohl postavit na běžném formátu Android XML, který využívají i aplikace psané v Javě.

V rámci .NET existuje samozřejmě i možnost vývoje cloudové aplikace na technologii ASP.NET Core, které jsem využil. Tato aplikace opět může sdílet a také sdílí společný model v podobě sdíleného projektu. Všechny potřebné třídy z tohoto projektu se zkompilují do aplikace Xamarin i ASP.NET Core, takže výsledný program je stejný, jako kdyby dané třídy byly umístěné přímo v kompilované aplikaci. (13) Android aplikace si pak s tou cloudovou vyměňuje informace pomocí RESTful aplikačního rozhraní.

Celé řešení jsem vyvíjel v prostředí Microsoft Visual Studio 2019 (verze 16.1.0) s rozšířením Xamarin (verze 16.1.0.542) a Xamarin.Android SDK (verze 9.3.0.22) v jazyce C# (verze 7.3). Cloudová aplikace je postavená na aktuální platformě .NET Core 2.2, mobilní aplikace je cílena na Android 9.0 Pie (API 28) s podporou starších verzí až k Androidu 5.0 Lollipop (API 21). To znamená, že aplikace může být spuštěna na 89,3 % zařízení s Androidem. (14)



Obrázek 15 – Tržní podíl jednotlivých verzí platformy Android, stav k 7. 5. 2019 (14)

4.2 Řešení sady Visual Studio

Mé řešení v rámci Visual Studia obsahuje celkem 4 projekty. Android aplikaci, Cloudovou aplikaci, Sdílený projekt a Testovací projekt, který obsahuje softwarové testy. Základní obor názvů je pro všechny testy společný a je jím `Schedules`. V následujících podkapitolách části Implementace se podrobněji rozepíši o tom, co obsahují. Oproti části Návrh jsem se rozhodl text nestrukturovat podle jednotlivých projektů, ale spíše podle funkcionalit, jelikož řešení jsou i díky sdílenému projektu často velmi propojená.

4.2.1 Android aplikace

Mobilní aplikace na platformě Xamarin.Android je velmi podobná té psané nativně v jazyce Java, proto se ve své práci zaměřím spíše na specifika mého řešení než na popis funkcí platformy. Má aplikace se skládá z 8 aktivit v oboru názvů `Schedules.Activities`, mezi nimiž se nachází i vstupní bod `MainActivity`. V kódu se třída představující aktivitu značí atributem `[Activity]` s volitelnými parametry specifikujícími například název, režim spouštění nebo téma jejího vzhledu. Tento atribut slouží k automatické generaci souboru AndroidManifest.xml, který by jinak bylo nutné vyplnit ručně.

Dále v projektu nalezneme 6 fragmentů, z nichž 5 tvoří hlavní obrazovky 3.1.1-3.1.5 umístěné zobrazované uvnitř `MainActivity`. V oboru názvů `Schedules.Adapters` se nachází adaptéry sloužící k zobrazování obsahu prvků `RecyclerView` (seznamy) a `ViewPager` (fragmenty) a v oboru `Schedules.BackgroundServices` máme službu `DelayService`, která se stará o odesílání informací potřebných k určování zpoždění autobusů. V neposlední řadě v podsložkách složky Resources jsou zdroje pro uživatelské rozhraní jako obrázky, ikony, nabídky, styly, lokalizované řetězce a hlavně rozložení – vše shodné s nativním přístupem, rozdíl je akorát v příponě souborů s rozložením `.axml` namísto `.xml` a v použití v kódu; je třeba k identifikátorům těchto zdrojů přistupovat v souladu s konvencemi jazyka C# přes celý název Resources s využitím velkého písmena na začátku názvu třídy, tedy například `Resources.Layout.bus`, namísto `R.layout.bus`.

Komunikace s cloudovou aplikací je zajištěna pomocí statické třídy `OnlineDataService`. Zde se nachází podpůrné metody pro zasílání požadavků a vracení jejich výsledků.

4.2.2 Cloudová aplikace

Cloudová aplikace obsahuje pouze 8 tříd, z nichž třídy `Program` a `Startup`, jak název napovídá, slouží k zajištění základního běhu. Pro každý navržený modul se zde pak nachází jeden kontroler (obor názvů `Schedules.Controllers`) a jedna datová služba (obor `Schedules.DataServices`).

Kontrolery přijímají požadavky z mobilní aplikace přes RESTful rozhraní a vyřizují je za pomocí datových služeb, které se starají o uchovávání dat.

4.2.3 Sdílený projekt

Sdílený projekt obsahuje zejména třídy využívané pro uchovávání dat o jízdních řádech, zpožděních i mimořádnostech. V oboru názvů `Schedules.Models` se nachází podobory `Requests` a `Responses` s třídami využívané pro komunikaci mezi aplikacemi. V oboru `Schedules.Helpers` nalezneme pomocné třídy. Kromě toho ve statické třídě `Settings` z oboru názvů `Schedules` jsou uchovávány takříkajíc natvrdo zadané konfigurační hodnoty.

4.3 Zpracování dat o jízdních řádech

V části 3.2.1 jsem se zmínil o tom, že data o jízdních řádech jsou pro aplikaci stěžejní a je na nich vše postaveno. Proto se jim budu věnovat nejdříve.

4.3.1 Dostupnost dat

Jízdní řády celé pravidelné veřejné dopravy jsou v České republice shromažďovány v Centrálním informačním systému o jízdních řádech (CIS JŘ). Ten spravuje na základě smlouvy se státem již několikrát zmíněná společnost CHAPS, spol. s r. o., která se dlouhodobě snaží komplikovat otevírání dat. Od doby, kdy jí stát přikázal data zveřejňovat ve strojově čitelné podobě, jsme mohli být svědky mnoha obstrukcí. Nejznámější je asi zveřejnění jízdních řádů jako tabulek ve formátu XLSX, tedy pro tabulkový procesor Microsoft Excel, které trvalo do konce srpna 2015. Dnes jsou jízdní řády autobusové dopravy dostupné přes protokol FTP na adrese ftp://ftp.cisjr.cz/ ve formátu nazvaném JDF. Ten byl vyvinut touto společností jako specifikace formátu CSV. Jde tedy o několik textových souborů, které jsou ale bohužel pro každou linku zabaleny v archivu ZIP a stejně tak jsou všechny tyto balíky uvnitř jednoho velkého ZIPu, což nápadně připomíná pokus o další znepříjemnění práce s těmito daty. (7)

Pro úplnost doplním, že CHAPS byl na konci října 2017 koupen dceřinou společností Českých drah, ČD-Informační systémy a tím fakticky přešla do vlastnictví státu. Od té doby se ale v otevřenosti dat nic nezměnilo a zveřejňováno je stále pouze nutné minimum, které vyžaduje zákon. (15; 16)

Dále jsou zapotřebí nějaké poziční informace, v nejlepším případě by to měla být data přesně popisující trasu linky. Takováto celorepublikově zveřejněna bohužel nejsou, přestože lze předpokládat, že existují. Napovídá tomu i služba Map IDOS, která zobrazuje trasy linek nad mapou tak, že ne vždy trasa zcela překrývá silnici. Pro účely bakalářské práce mohu data nasimulovat, případně využít méně přesný způsob odhadování trasy na bázi souřadnic jednotlivých zastávek. Bohužel ani tento přístup by se neobešel bez problémů, CHAPS totiž nezveřejňuje ani tyto informace. Proto jsem se rozhodl sáhnout po datech náhradních, která pro své území zveřejňuje Regionální organizátor Pražské integrované dopravy (ROPID). (17) Bohužel, či možná pro cestující naštěstí, jak jsem se zmiňoval v závěrech analýzy (2.2), většina spojů v PID již má online data o zpoždění veřejně k dispozici, proto v případě mé práce jde zejména o demonstraci technologie odhadování zpoždění na reálných datech.

4.3.2 Formát dat

ROPID poskytuje data podle mezinárodní specifikace GTFS Static spravované společností Google. Navíc využívá rozšíření Trip-to-trip transfers, které umožňuje popisovat garantované přestupní vazby, a poskytuje podrobnější strukturovaný seznam zastávek a zastávkových sloupků ve formátu XML a Json. Jelikož přestupům se ve své práci nevěnuji, tak jsem zmíněné rozšíření nevyužil, nicméně podrobné informace o zastávkách jsou pro moji aplikaci užitečné například při zobrazování mapy spojení.

4.3.3 Stahování a uchovávání dat

Jízdní řády si stahuje ze zdroje ROPIDu cloudová aplikace při svém spuštění. Nejprve proběhne kontrola aktuálnosti již v minulosti stažených dat; pokud jsou k dispozici konzistentní data mladší než 1 den, jsou použita, v opačném případě jsou stažena nová. Aktualizace těchto dat za běhu aplikace je provizorně řešena externí funkcí zasílající jednou denně požadavek HTTP GET na adresu `api/schedules`. Data jsou uchovávána v operační paměti v modelu sdíleném s Android aplikací popsaném v části 4.3.4.

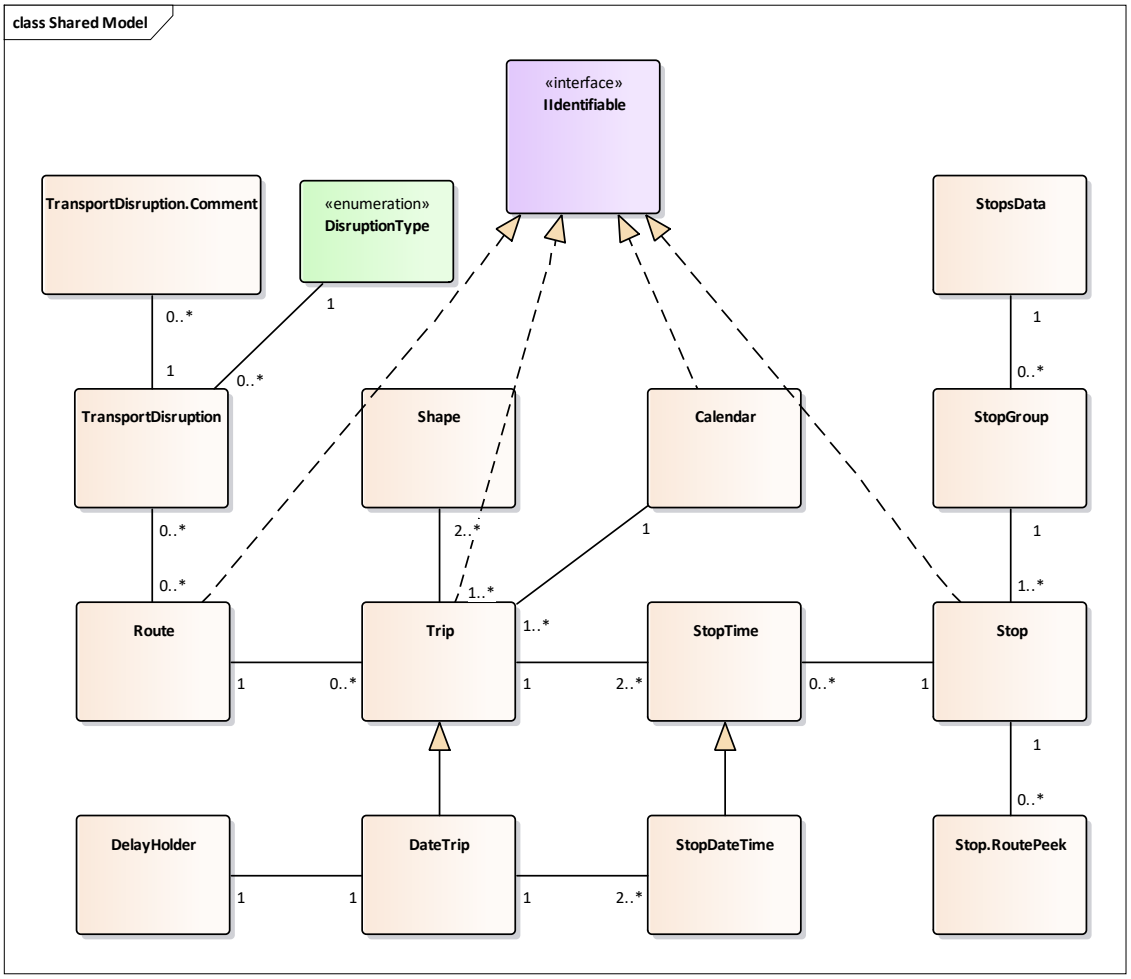
Stažení zajišťuje třída `SchedulesDataService`. V poměrně dlouhé metodě `UpdatePidData()` jsou po stažení nejprve deserializována data o zastávkách z formátu Json a poté GTFS data postupně extrahována z archivu, navazována na zastávky a propojována mezi sebou.

4.3.4 Sdílený datový model

Jak je vidět na diagramu (Obrázek 16 na následující stránce), datový model Sdíleného projektu obsahuje 11 tříd nutných pro uchovávání dat o jízdních řádech. To jsou všechny, které jsou umístěny ve třech pravých sloupcích a z levého sloupce ještě třída `Route`. Většina z těchto tříd a jejich vlastností vychází z formátu GTFS, základ tříd v pravém sloupci diagramu je deserializovaným ekvivalentem specifikace organizace ROPID, o kterém jsem se zmiňoval výše. (18; 17)

Základem datového modelu je třída `Trip`, která reprezentuje autobusový spoj v jízdním řádu. Její specializace `DateTrip` představuje konkrétní spoj pro zadaný operační den. Ten může být odlišný ode dne výjezdu z výchozí stanice zejména u nočních linek. `Route` značí linku daného spoje a `Calendar` obsahuje informace o tom, pro které dny je daný `Trip` platný. `Shape` pak reprezentuje bod trasy spoje a `StopTime` zastávku, resp. přesněji zastavení, na cestě `Trip`u – ekvivalentně `StopDateTime` pro `DateTrip`. Třída `Stop` představuje zastávkový sloupek a `StopGroup` celou zastávku s unikátním názvem. Stručný náhled na linky obsluhující danou zastávku zajišťuje `Stop.RoutePeek` a `StopsData` je pouze obalující třída vycházející z formátu ROPIDu.

Třídy `DelayHolder` a `TransportDisruption` slouží k uchovávání dat o zpoždění a mimořádnostech a rozhraní `IIdentifiable` implementují třídy obsahující jedinečný identifikátor. To pomáhá například při vytváření slovníků.



Obrázek 16 – Diagram tříd Sdíleného projektu

4.3.5 Předávání dat z cloudu do telefonu

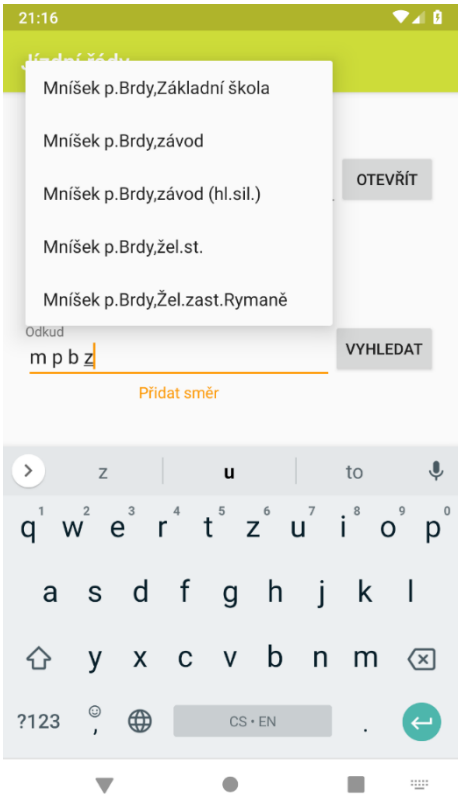
Metody zodpovědné za výměnu informací najdeme, jak již bylo řečeno, v projektu Cloudová aplikace, u jízdních řádů je to ve třídě `SchedulesController`. Všechny v této třídě (kromě konstruktoru) slouží k přebírání HTTP GET požadavků, to značí nad nimi umístěný atribut `[HttpGet(string)]`. Řetězec uvedený jako parametr představuje šablonu adresy URL, pomocí které se z adresy vyčtou jednotlivé parametry metody kontroleru. Do složených závorek se uvádí název parametru. (19)

Při vracení hodnot jako odpověď na požadavek probíhá automaticky serializace do formátu Json. Aplikace je nastavena tak, že při konfiguraci řešení Debug formátuje Json výstup s odsazením pro lepší a při konfiguraci Release nikoli, aby byla měla data co nejmenší velikost. Za účelem snížení velikosti přenášených dat jsou některá pole při serializaci ignorována, to zajišťuje atribut `[JsonIgnore]`. Dále jsem také vytvořil třídu `IgnorePropertiesContractResolver`, která při umístění do nastavení serializéru zařídí, že jsou do jejího konstruktoru zadané názvy polí a vlastností ignorovány, takže lze ignorování zapínat a vypínat dynamicky podle kontextu.

NAŠEPTÁVÁNÍ

V mobilní aplikaci se na obrazovce Vyhledávání (třída `SearchFragment`) nachází dvě textová pole určená pro zadávání názvu zastávek a jedno pro linky. Jelikož jde o pole vyhledávací, je na místě, aby měla funkci našeptávání, takže jsem využil element `AutoCompleteTextView`. Protože zejména v případě zastávek jde o větší objem zdrojových dat, která se ještě poměrně často (jednou denně) mění, implementoval jsem našeptávání na serveru, takže mobilní aplikace vyšle v případě zastávek požadavek pomocí metody `OnlineDataService.DownloadStopSearchResultsAsync(string)`, jehož výsledky zobrazí jako navržené zastávky pro aktuální dotaz.

Na straně cloudové aplikace v případě vyhledávání linek kontroler pouze vyfiltruje linky, jejichž označení či trasa začíná zadaným dotazem. Využívá se zde má rozšiřující metoda pro `string` s názvem `ConvertSpecialAndUpperCharacters(…)`, která odstraní diakritiku, speciální znaky a převede velká písmena na malá. V případě zastávek volá kontroler metodu `SearchStopGroup(string)` v datové službě. Ta nejprve vyfiltruje výsledky algoritmem nazvaným `cleverfilter`. Ten využívá metody `WordsStartWith(this string)`, která slouží pro vyhledávání, jaké zavedlo v poslední době mnoho vyhledávačů v jízdních řádech, kde například dotaz „m p b z“ vrátí jako výsledek mj. zastávku *Mníšek pod Brdy, železniční stanice* (Obrázek 17). Výsledky, které nalezne `cleverfilter`, se množinově sjednotí s výsledky algoritmu stejného jako v případě linek s tím, že ty *chytré* se zobrazují na prvních místech.



Obrázek 17 – Našeptávání zastávek

VYHLEDÁVÁNÍ ODJEZDŮ A SPOJENÍ

Pravděpodobně hlavní funkcí v rámci jízdních řádů je vyhledávání odjezdů a spojení. Pro rychlé vyhledávání odjezdů jsem vytvořil kolekci `DupSortedList<T, TComp>`. Je postavená na základě seznamu, který obsahuje seřazené prvky. Typ `TComp` představuje jakýsi klíč, který je z položky seznamu vybrán selektorem předaným třídě v konstruktoru. Podle něj jsou položky řazeny. Klíč to však není v pravém slova smyslu, jelikož se v kolekci může vyskytovat vícekrát; proto jsem také nevyužil C# kolekce `SortedList<TKey, TValue>` nebo `SortedDictionary<TKey, TValue>`, které toto neumožňují. V mém případě jde o užitečnou vlastnost, jelikož do této kolekce jsem umístil objekty `StopTime` s časy odjezdů jako klíči.

Hledání odjezdů je v datové službě cloudové aplikace implementováno metodou `FindDepartures(int,…)`, jejíž první parametr je číslo zastávky z centrálního informačního systému (CIS). Nejprve se pomocí binárního hledání nalezne nejnižší index odpovídající

položce se zadaným časem v metodě `FindLowestIndex(TimeSpan)`, která je z třídy `DupSortedList`. Následně algoritmus prochází všechny časy odjezdů maximálně 3 dny dopředu (v případě, že by výsledků byl malý počet), dokud nenarazí na příslušný počet odpovídajících výsledků. Výsledky se vrací v podobě typu `TripWithStop`, který akorát osahuje daný `DateTrip` a číslo (indexované od 1) pořadí zastávky na trase spoje, ze které to je odjezd. Pro přenos do mobilní aplikace se používá typ `IncompleteTripWithStop`, který vychází z `TripWithStop`, akorát jeho název značí, že při serializaci z něj budou odebrány některé informace – konkrétně jízdní řád (kromě odjezdové zastávky) a trasa spoje.

Algoritmus pro hledání přímých spojení funguje na podobném principu. Nejprve vyhledá ze zadané výchozí stanice odjezdy linek, které zastavují i v zadané cílové stanici, a poté ověří, zdali vyhledané spoje opravdu na své budoucí trase v cílové zastávce zastaví.

Obrázek 18 – Vyhledávání spojení

4.3.6 Změny v uživatelském rozhraní

Oproti návrhu (3.1.1) doznala obrazovka Vyhledávání několik změn na základě problémů odchycených při testování (3.3), které bych rád zde přiblížil. V první řadě byla sloučena část vyhledávání přímých spojů s vyhledáváním odjezdů do jedné, ve které se nachází jedno pole a po klepnutí lze přidat směr cesty přímého spoje. Uživatelé totiž rozdělení na tři části považovali za matoucí. Dále byla vzhledově upravena a přesunuta možnost nastavení data a času odjezdu tak, aby lépe odpovídala svému účelu, a pro nedostatek času odebrána funkce vyhledávání příjezdů, jelikož pro uživatele nepřináší zásadní užitek a pro vývojáře představovala větší množství času. Do budoucna navrhuji po stránce UI funkci implementovat pomocí elementu `Spinner`.

4.4 Historie a úložiště v mobilní aplikaci

Obrazovka Poslední zobrazuje historii otevřených zastávek a autobusů. (Obrázek 19Obrázek 25) Po stránce uživatelského rozhraní je dle návrhu velmi podobné obrazovce Nejbližší, takže se rozhraní věnuji až v části 4.5. Zde se zaměřím na pozadí specifické pro ukládání dat o historii. Ta pochopitelně musí být uchovávána i po ukončení aplikace. Zařídit to je možné uložením souboru do paměti telefonu, případně využitím rozhraní `SharedPreferences` obsahujícího hodnoty v základních datových typech uložené pod klíčem v podobě řetězce. (20)

Zvolil jsem druhou možnost, jelikož je přímočařejší. Data o autobusech a zastávkách sice nejsou základní datový typ, ale je možné je převést na řetězec opět serializací. Za tímto účelem jsem ve statické třídě `OfflineDataService` naprogramoval generické metody `GetHistory<T>(…)` a `UpdateHistory<T>(…)`. Záznamy historie se ukládají pod klíčem v podobě tiků odpovídajících času uložení a klíče získaného funkcí `keySelector` oddělených dvojtečkou, hodnota je opět ve formátu Json. Seznam klíčů je navíc uložen jako `StringSet` pod klíčem „keys". Pro každý druh historie existuje přetížení metody `UpdateHistory(…)` a ekvivalentní metoda pro její získání. Každý druh se také ukládá do zvláštního souboru (mají jiný název `SharedPreferences`).

Kromě toho se ve službě `OfflineDataService` nachází metody pro ukládání aktuálního autobusu, výstupní zastávky, jména nebo komentování mimořádností, takže slouží jako středobod pro správu perzistentních dat, která jsou umístěna lokálně v telefonu.

4.5 Nejbližší zastávky a autobusy

Přehled nejbližších zastávek je implementován v třídě `NearestFragment` a adaptéru `RecentNearestStopsAdapter`. Ten je také, jak název napovídá, sdílen s obrazovkou Poslední. Implementuje rozhraní `ILocationListener`, díky čemuž je využíván jako cíl pro aktualizace polohy od systému Android skrze `LocationManager`. To se využívá pro výpočet vzdálenosti zastávky od polohy uživatele. Obdobně mají svůj adaptér nejbližší a poslední autobusy, ve kterém se nachází vlastnost `IsNearest` s logickou hodnotou, která značí, zdali se jedná o adaptér pro nejbližší či poslední spoje.

Uvnitř fragmentu se v metodě `OnAttach(Context)`, která se volá při jeho otevření, vyžádá získávání informací o poloze zařízení, v metodě `OnDetach()` se naopak toto získávání ukončí. Při získání aktualizace se vždy na novém vlákně stáhnou nejbližší zastávky a autobusy tak, aby nebylo blokováno vlákno hlavní s uživatelským rozhraním.

Na straně cloudové aplikace je toto řešeno v datové službě `SchedulesDataService` metodami `FindNearestStopGroups(float, float)` a `FindNearestTrips(float, float, int)` přijímajícími jako parametry zeměpisnou šířku a délku. První zmíněná pouze vrátí zastávky seřazení pomocí C# rozšíření LINQ. Ta druhá taktéž vyhledá nejbližší zastávky, z nich vybere ty, které jsou ve vzdálenosti do 500 m. Jestliže žádná taková není, vezme první v seznamu. Z každé z nich vyhledá odjezdy začínající aktuálním časem mínus 2 minuty jakožto tolerancí pro právě odjíždějící spoje. Spoje vrací opět v podobě objektu `IncompleteTripWithStop`.

K výpočtu vzdáleností se využívá se využívá známý vzorec Haversine. Jeho implementace není součástí .NET Core, takže jsem ji vložil do pomocné třídy `Geo`. (21)

4.6 Mimořádnosti

Uživatelské rozhraní k mimořádnostem patří mezi ty složitější v mé aplikaci. Základem je `DisruptionsFragment`, který je zobrazován v hlavní aktivitě, `DisruptionActivity` pak zobrazuje detail konkrétní mimořádnosti. Součástí aplikace je ještě `DisruptionAddActivity` sloužící pro přidávání nových mimořádností, tomu se věnuji níže, a také statická třída `DisruptionsAdapterCore` s jednou metodou, kterou využívá `GenericRecyclerAdapter<T>`. Ten jsem vyvinul jako obecnou třídu, díky níž není nutné pro každé `RecyclerView` vytvářet novou třídu. Stačí obecnému adaptéru přiřadit položky, odkaz na rozložení a metodu `OnBindViewHolder(RecyclerView.ViewHolder, T)`. Metoda z `DisruptionsAdapterCore` je využívána jak pro zobrazení mimořádností uvnitř jejich fragmentu, tak na stránkách Přehled jízdy autobusu a Můj autobus. Na nich se zobrazují mimořádnosti, jejichž zadaná ovlivněná linka se shoduje s linkou zobrazeného autobusu.

Fragment a aktivita pro mimořádnosti jinak neobsahují žádné významnější technické kousky. Zmíním pouze, že v `DisruptionActivity` (Obrázek 20) se nachází Google mapa z Google Play APIs s polohou mimořádnosti a že autor může skrze panel nástrojů v aktivitě nebo dlouhým podržením v seznamu uvnitř fragmentu přes kontextovou nabídku mimořádnost smazat.

V seznamech událostí jsou položky řazeny podle skóre hodnocení, tedy kladné hlasy mínus záporné hlasy. Toto řazení probíhá již v cloudové aplikaci, s níž ta mobilní komunikuje skrze kontroler `DisruptionsController` a správa dat probíhá pomocí služby `DisruptionsDataService`. Při zaslání požadavku GET na adresu `api/disruptions` (výpis všech mimořádností) kontroler nejprve smaže všechny mimořádnosti, které nebyly žádným uživatelem pozitivně hodnoceny za posledních 24 hodin (kromě těch, které byly v posledních 24 hodinách nahlášeny). To je velmi jednoduchý mechanizmus pro průběžné mazání neaktivních mimořádností, které nesmaže autor, tak, aby se uživatelům nezobrazovaly irelevantní položky. Do budoucna je velmi vhodné tento mechanizmus zpřesnit například podle počtu otevření události a změnit z mazání na skrývání.

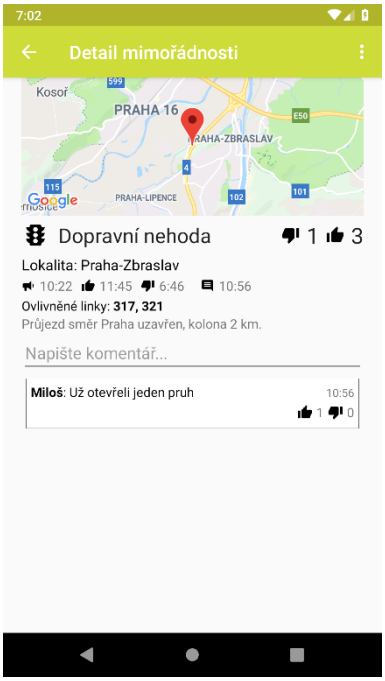
4.6.1 Přidávání mimořádností

Přidávání v mobilní aplikaci probíhá, jak bylo naznačeno, skrze aktivitu `DisruptionAddActivity`. Ta byla naimplementována s drobnými změnami oproti návrhu. Tlačítko pro vložení mimořádnosti bylo přesunuto do aplikační lišty a lokalita se vybírá klepnutím na mapu. Proces přidávání probíhá zasláním HTTP POST požadavku s mimořádností typu `TransportDisruption`, jejíž pole `Author` obsahuje Instance ID, tedy jedinečný identifikátor poskytnutý službami Google Play zejména pro účely komunikace s cloudovými službami. (22) Pomocí tohoto identifikátoru taktéž rozlišuji autory komentářů a hodnocení mimořádných událostí.

Aktuálně jsou na straně cloudu mimořádnosti uchovávány v operační paměti s tím, že se při každé změně obsahu (kromě hodnocení) zálohují na místní disk ve formátu Json. Tato záloha

je kontrolována při spuštění cloudové aplikace a případně obnovena. Je jasné, že tento přístup si do budoucna žádá změnu například na ukládání do relační databáze.

Obrázek 19 – Obrazovka Poslední



Obrázek 20 – Detail mimořádnosti

Obrázek 21 – Detail zastávky

4.7 Můj autobus

Z hlavního uživatelského rozhraní se budu podrobněji věnovat ještě klíčové obrazovce Můj autobus a její sestře Přehled jízdy autobusu. Sdílí společné rozložení definované v souboru `bus.axml`, ale první zmíněná představuje `BusFragment`, druhá je `BusActivity`.

Středobodem obou obrazovek je `RecyclerView`, které střídá dva adaptéry. `TripShortScheduleAdapter` pro zkrácený seznam zastávek a `TripScheduleAdapter` pro kompletní jízdní řád spoje. Druhý jmenovaný je o něco jednodušší. Oba přijímají jako parametr autobus (`DateTrip`), číslo výstupní zastávky (`int?` – číslo s podporou hodnoty `null` pro případ nezadané výstupní zastávky) a volitelně informace o zpoždění (`DelayInfo` – podrobněji rozebrané v kapitole 4.8). Druhý adaptér ale pouze zobrazuje jízdní řád a případně do něj vloží řádek s aktuální polohou, jestliže se autobus nenachází v zastávce. Pokud autobus v zastávce je, označí se zastávka tučně a symbolem `ic_location_on`. Adaptér pro kratší seznam zastávek provádí zásadní práci v metodě `Initialization()`. V ní nejprve v případě, že není k dispozici informace o zpoždění, odhadne možnou polohu na základě aktuálního času a jízdního řádu taktéž v podobě objektu `DelayInfo`. Následně určí pozice adaptéru (index 0-4) pro jednotlivé položky jako výchozí zastávka, konečná zastávka, aktuální, předchozí a následující zastávka nebo řádek s informací o zpoždění. Nakonec rozmístí oddělovače v podobě tří teček na místa, kde byly oproti kompletnímu jízdnímu řádu nějaké zastávky vynechány. Výsledný seznam tedy má 2 až 7 položek (včetně oddělovačů).

Stahování informací o zpoždění a mimořádnostech probíhá opět v novém vlákně vytvořeném v metodě `OnResume()` pravidelně každých 30 vteřin a je možné jej také vyvolat ručně potažením prstem shora dolů. Poslední známá poloha autobusu a mimořádnosti se také zobrazují (zeleně a oranžově) na mapě trasy spoje, kterou je možno otevřít klepnutím na tlačítko na panelu nástrojů. Mapa se zobrazuje v aktivitě `MapActivity`. (Obrázek 23)

4.7.1 Specifika aktivity

`BusActivity` má oproti fragmentu jedno specifikum, kterým je plovoucí tlačítko `FloatingActionButton`. (23 str. 159) (Obrázek 22) To slouží dle návrhu k označení nástupu do autobusu. Tlačítko se v aktivitě zobrazí pouze pokud autobus je dle jízdního řádu právě na své trase s tolerancí -5 minut a +2 hodiny.

4.7.2 Specifika fragmentu

Fragment je naopak specifičtější výrazněji. To je dáno tím, že jeho stav musí být perzistentní a vyžaduje více informací. Tou informací je zejména uživatelova výstupní zastávka. Při prvním spuštění fragmentu s daným autobusem (obvykle po kliknutí na plovoucí tlačítko v aktivitě) je uživatel vyzván pomocí `AlertDialogu` k výběru výstupní zastávky. Dialog se volá metodou `SelectMyDestinationDialog(bool)` a je zkrácený pro lepší přehlednost pouze na budoucí zastávky (zohledňuje zpoždění spoje), ale v patičce má i možnost zobrazit seznam zastávek kompletní. Dialog se nezobrazuje, jestliže je z kontextu patrné, v jaké zastávce bude uživatel vystupovat. To se stane v případě, že spoj vyhledá přes vyhledávání přímých spojení se zadáním cílové zastávky do vyhledávače. (23 str. 160)

Pro perzistenci informací o autobusu a výstupní zastávce se opět používají, jak již bylo psáno, `SharedPreferences` skrze `OfflineDataService` jako v případě 4.4.

4.7.3 Změny oproti návrhu

Obrazovky s autobusem doznaly oproti návrhu trochu více změn. Nejprve bylo licenční číslo linky odebráno a nahrazeno textem „spoj č.“, aby byla informace srozumitelnější i pro uživatele méně zběhlé v jízdních řádech. Jako na ostatních obrazovkách, jsem nahradil tlačítko aktualizovat gestem v rozložení `SwipeRefreshLayout`. To navíc nebyla jediná změna v panelu nástrojů, ještě ve fragmentu jsem přesunul nabídku skrytou pod tlačítkem autobusu do klasické doplňkové nabídky ukryté pod ikonou se třemi tečkami a zachoval.

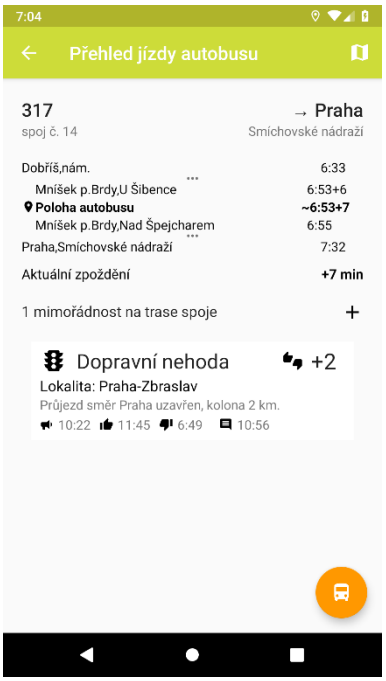
Na základě zpětné vazby byla na obrazovky také umístěna explicitní informace o aktuálním zpoždění, jestliže je poslední dostupná informace maximálně 5 minut stará. Odebrány tedy byly piktogramy nacházející se pod cílovou zastávkou uvnitř `BusFragmentu`.

4.8 Zpoždění

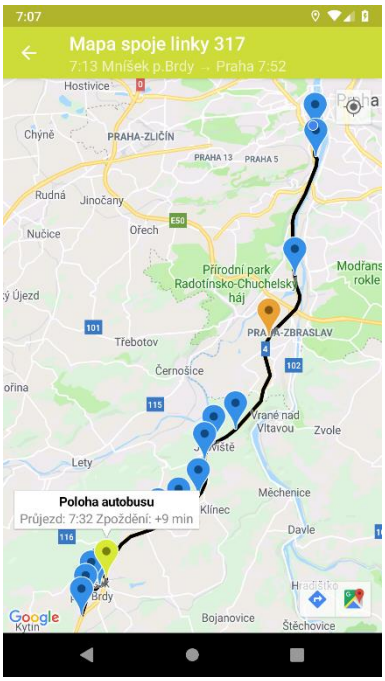
Jádro celého systému spočívá v pravidelném odesílání polohy z mobilní aplikace do té cloudové. Děje se tak jednou za 15 sekund ze služby `DelayService`. Ta je v rámci platformy

Android tzv. službou na popředí, tedy procesem, který nemá uživatelské rozhraní, běží i na pozadí aplikace, ale informuje o svém běhu skrze trvalou notifikaci (*ongoing notification*). (23 str. 258; 24) (Obrázek 24) Služba je spouštěna zároveň s `BusFragmentem`.

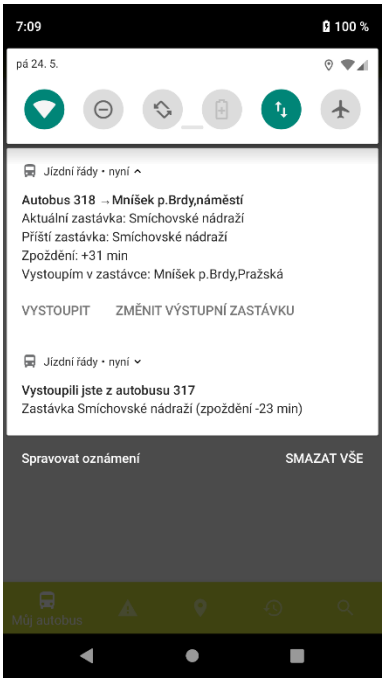
Po spuštění se přihlásí opět skrze `LocationManager` o aktualizace polohy a při jejich přebírání zjistí pomocí vestavěné třídy `Geocoder` název místa, kde se uživatel nachází. Poté vytvoří záznam `DelayRecord` obsahující informace o poloze, autobusu a Instance ID. To je připravené k odeslání do cloudové aplikace. Před odesláním ještě zkontroluje, zdali se uživatel nenachází dále než 1 km od trasy autobusu. To slouží jako ochrana před zneužíváním. Tolerance ve výši 1 km byla zvolena tak, aby počítala s případným menším operativním odklonem od trasy nezaneseném do jízdního řádu. V případě, že se uživatel nachází příliš daleko, služba je ukončena, spoj odebrán z obrazovky Můj autobus a uživatel je o všem informován prostřednictvím `Toast` upozornění a standardní notifikace. Po ověření následuje odeslání informace na server a po odeslání se zkontroluje, zdali již uživatel není ve své cílové zastávce nebo případně za ní, a aktualizuje se trvalá notifikace. Ta v průběhu cesty ukazuje mj. aktuální výši zpoždění a následující zastávku. Při dosažení cílové zastávky je služba ukončena taktéž s upozorněním uživatele.



Obrázek 22 – Přehled jízdy autobusu



Obrázek 23 – Mapa spoje



Obrázek 24 – Notifikace o zpoždění

V cloudu se data ukládají objektů třídy `DelayHolder`. Jeden takový objekt existuje pro jeden unikátní `DateTrip`. Obsahuje zejména údaje o tom, kdy a jaký uživatel projel kterou částí trasy. To obsahuje je slovník s objekty `DateTime` jako hodnotami a `ValueTuple (double TripDistance, string InstanceID)` jako klíči. Časy průjezdů se určují podle času doručení požadavku na server. Existuje zde sice riziko vzniku nepřesností při pomalejším spojení, avšak ta zanedbávám, jelikož druhou možností je řídit čas podle hodin telefonu,

které nemusí být vždy přesné, a navíc by bylo možné tímto způsobem data podvrhnout. Objekty typu `DelayHolder` se uchovávají ve třídě `DelayDataService` opět ve slovníku, tentokrát s klíči identifikujícími `DateTrip` – tedy identifikátor `Tripu` z GTFS a operační den.

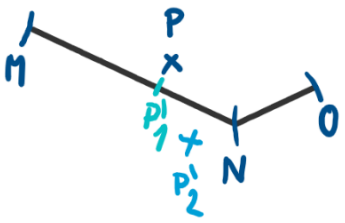
Při vyžádání dat o zpoždění skrze požadavek obsahující identifikátory spoje se kontroler pokusí získat příslušný `DelayHolder` a na jeho základě vytvoří objekt typu `DelayInfo`, který obsahuje data zpracovaná pro snadné čtení v mobilní aplikaci. Jejich základem je slovník `StopDelays` obsahující výše zpoždění na základě sekvenčního čísla zastávky.

4.8.1 Metody sloužící k určení polohy a zpoždění

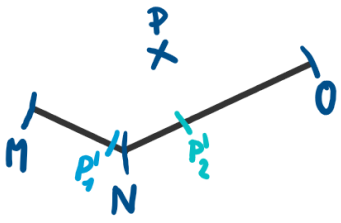
Pro tyto výše popsané funkce se využívá několik metod, které nyní přiblížím.

Stěžejním prvkem jsou metody umístěné ve třídě `Trip`, zejména metoda **`EstimatePositionOnRoute(float, float)`**. Tato metoda přebírá zeměpisné souřadnice a promítá je pomocí metod v pomocné třídě `Geo` na trasu spoje. Děje se tak tím, že nejprve vybere nejbližší bod trasy a následně promítne pomocí ortogonální projekce zadanou polohu na přímky mezi nejbližším a následujícím bodem trasy a mezi nejbližším a předchozím bodem trasy. (25 str. 37) Jestliže se jeden z průmětů nachází přímo na úsečce mezi těmi body, použije se daný průmět jako odhadnutá poloha autobusu na trase. Jestliže se oba průměty nachází na úsečkách, použije se ten, jehož vzdálenost od zadané polohy (zařízení) je menší. V případě, že se ani jeden průmět nenachází na úsečce, jako odhadnutá poloha se použije nejbližší bod trasy.

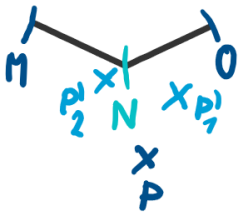
Zde jsem připravil ilustraci tohoto problému. N je nejbližší bod trasy, M je předchozí bod a O je následující. P značí polohu zařízení, P_1' průmět na přímku s procházející předchozím bodem a P_2' průmět na přímku procházející následujícím. Zelenějším odstínem je vyznačen bod značící odhadnutou polohu autobusu na trase.



Obrázek 25 – Průmět na 1 úsečku



Obrázek 26 – Průmět na 2 úsečky



Obrázek 27 – Průmět mimo úsečky

Výsledek této metody lze použít jako parametr pro **`GetTraveledDistance(Vector2)`**. Ta prochází jednotlivé body trasy. Ty obsahují informaci o ujeté vzdálenosti, takže není problém ji na jejich základě dopočítat i pro zadané souřadnice ležící na oné trase. Přesně opačným způsobem funguje i poslední metoda z třídy `Trip` **`GetPositionForDistance(double)`**, která na základě ujeté vzdálenosti vrací souřadnice na trase spoje.

V odvozené třídě `DateTrip` je pak umístěna metoda **`EstimateScheduledTransit(double)`**. Tato metoda odhaduje čas pravidelného průjezdu na daném kilometru spoje. Vyhledá předchozí a následující zastávku kolem zadaného místa, vypočte z jízdního řádu průměrnou rychlost jízdy autobusu v tomto úseku a podle ní pak vypočte, v kolik hodin projede autobus zadaný kilometr.

Konečně pak konstruktory třídy `DelayInfo` vytvářejí finální výsledek. Konstruktor **`DelayInfo(DelaryRecord, DateTrip)`** slouží k určení zpoždění pouze na základě jednoho záznamu o poloze a využívá se pro aktualizaci trvalé notifikace v mobilní aplikaci. Naproti tomu konstruktor **`DelayInfo(DelayHolder)`** využívá metodu z třídy `DelayHolder` s názvem **`EstimateStopDelays()`**, která zpracovává veškeré nasbírané záznamy k danému autobusu do slovníku obsahujícího zpoždění pro každou zastávku. Tato metoda nejprve rozdělí záznamy podle zdroje, tedy uživatelů. Z těch pak jednotlivě vypočte zpoždění pro každou zastávku, kterou uživatel projel. Pro výpočet jsou zatím využívány vždycky pouze maximálně dva nejbližší záznamy z každé strany zastávky. Jestliže je nějaký záznam od zastávky blíže než 100 m u městských spojů nebo 150 m u meziměstských, je považován jako umístěný v zastávce. Na závěr jsou data od jednotlivých uživatelů pro každou zastávku zprůměrována a tím jsou vypočtena výsledná zpoždění.

4.9 První spuštění

Z testování prototypu vyplynulo (3.3.1), že by bylo vhodné uživatele při prvním spuštění krátce uvést do základních principů fungování aplikace. Vytvořil jsem tedy čtyři stručné úvodní obrazovky v podobě třídy `OnboardingFragment`, které se využívají v rámci `OnboardingActivity`: (26)

1. Uvítání a představení hlavního smyslu aplikace
2. Informace o způsobu získávání dat o zpoždění
3. Informace o fungování mimořádností
4. Upozornění na symbol polohy a pobídnutí uživatele

Aktivita se spustí při prvním startu aplikace i při každém dalším, jestliže naposledy nebyla dokončena. Stav dokončení se ukládá za pomoci výčtového typu `OfflineDataService.OnboardingResult` do paměti telefonu. Díky tomuto typu je tato funkce připravena na další rozšíření například pro zvláštní uvádění uživatelů do nové verze aplikace.

4.10 Poznámky a zajímavosti

4.10.1 Detail zastávky

Na obrazovce Detail zastávky se nachází `CollapsingToolbarLayout`, který ve své rozbalené podobě obsahuje náhled mapky. (27) (Obrázek 21) Nácestné zastávky u spojů vypsaných na této obrazovce jsou vybírány *jednořádkovým* LINQ algoritmem ve třídě `StopTime`, který je vybírá podle počtu přestupních linek:

```
Trip.StopTimes
    .Where(st => st.StopSequence > StopSequence)
    .SkipLast(1)
    .OrderByDescending(st => st.Stop.Group.Lines.Count)
    .Take(3)
    .OrderBy(st => st.StopSequence)
    .Select(st => st.Stop.Group.Name);
```

4.10.2 Lokalizace

Celá mobilní aplikace je lokalizována do češtiny a do angličtiny standardním způsobem, který se řídí nastavením jazyka v systému Android. (23 str. 116)

4.10.3 Cloud

Serverová aplikace je v době dokončení této práce zveřejněna ve službě Microsoft Azure na adrese https://schedules.azurewebsites.net/.

4.11 Softwarové testy

Testy pro .NET jsou implementovány v samostatném projektu. Ten má v mé práci název Testovací projekt s výchozím oborem názvů `Schedules.Tests`. Pokrýt jsem se rozhodl nejzásadnější třídy sdíleného datového modelu a datové služby cloudové aplikace.

Existuje několik testovacích frameworků, ze kterých jsem nakonec zvolil MSTest na platformě .NET Core. Jde o integrovaný framework od Microsoftu, který poskytuje veškerou základní funkcionalitu. Například oproti NUnit a dalším není tak rozšiřitelný, což ale není funkcionalita, kterou má menší práce vyžaduje. Naopak má podporu pro Azure DevOps (dříve VSTS), kde mám svůj kód umístěn. Nicméně jeho hlavní funkce sloužící ke kolaboraci stejně nevyužívám. V současné verzi Visual Studio 2019 existuje zabudovaná podpora nejen pro dva zmíněné frameworky, ale například i pro xUnit nebo WebDriver pro webové stránky. (28 str. 271)

5 TESTOVÁNÍ

5.1 Uživatelské testy

5.1.1 Skupina testerů

Pro testování výsledné aplikace jsem vybral 4 testery. Dva z nich již před půl rokem testovali prototyp (3.3), dva byli zcela noví. V každé z těchto skupin je jeden uživatel, který vlastní telefon s operačním systémem Android a jeden s telefonem se systémem iOS, takže se jedná o rovnoměrné zastoupení.

5.1.2 Testovací scénáře

Scénáře a úkoly jsem vybíral tak, aby se uživatelé při jejich plnění mohli dostat do všech zákoutí aplikace. Každý z nich měl nějaké očekávané „vývojářské" řešení, ale zdaleka ne každý byl ve finále splněn tím nejlepším způsobem. Snažil jsem se také zaměřit na části, které byly problematické při testování prototypu a byly ve fázi implementace nějak změněny.

Testování probíhalo v terénu, tedy v centru Prahy na autobusových linkách 176 v celé trase z Karlova náměstí do zastávky Stadion Strahov a 191 ze Stadionu Strahov na Anděl v časech mezi 11. a 16. hodinou. Linka 176 projíždí mj. po Rašínovo nábřeží, Jiráskově mostě a kolem Arbesova náměstí. V těchto místech z vlastní zkušenosti vím, že obvykle nabere několikaminutové zpoždění, což se při testování potvrdilo. Tím, že na lince 191 testování probíhalo jen v části trasy, jsem mohl otestovat zase jiné scénáře zejména při vyhledávání spojení a nástupu do vozidla. Všechny scénáře vyjma prvního byly zaměřeny na praktické situace v provozu:

1. Spustit aplikaci poprvé a projít onboarding
2. Najít vhodný autobus 176 z Karlova náměstí a nastoupit
3. Nahlásit mimořádnost (oprava silnice, posunutí zastávky) u Švandova divadla
4. Najít nejbližší autobusy z Cukráku na Smíchovské nádraží a vybrat si z nich vhodný
5. Zjistit aktuální zpoždění a zpoždění v zastávce Arbesovo náměstí
6. Najít autobus ze Strahova na Anděl a nastoupit
7. Uživatel se spletl, chtěl jet až do zastávky Na Knížecí
8. Uživatel se dal do řeči s kamarádem, který mu na Andělu oznámil, že vystupuje

Poslední zmíněný byl vždy přednesen tak, aby bylo patrné, že uživatel chce vystoupit s ním, a měl ukončit jízdu v aplikaci manuálně.

SIMULOVANÝ SCÉNÁŘ Č. 4

Kromě toho, že čtvrtý scénář byl testován v mezičase v průběhu jízdy autobusu, byla pro něj také nasimulovaná data o zpoždění a mimořádnostech na linkách 317, 318 a 321. Simulační režim je implementován do mobilní aplikace ve třídě `OnlineDataService` a je možné jej spustit dlouhým stisknutím textu „Žádný můj autobus" na prázdné obrazovce Můj autobus.

5.1.3 Průběh testování

Tabulka 3 – Stručný přehled chování testerů implementace

Úkol	Tester č. 1 Android, dříve #1	Tester č. 2 iOS, dříve #2	Tester č. 3 Android	Tester č. 4 iOS
1.	**Úspěch** Uživatel v pořádku, zobrazily se chybně noční spoje místo denních.	**Úspěch**	**Úspěch** Zpětně zjištěno, že jako jediný čte podrobně (přesto rychle).	**Úspěch**
2.	**Neúspěch** přes vyhledávání, ale v poli pro konkrétní spoj	**Částečný úspěch** Snažil se přes konkrétní spoj, ale pak zvolil Nejbližší.	**Úspěch** přes Nejbližší	**Částečný úspěch** Zkouší přes konkrétní spoj, nakonec spojení.
3.	**Částečný úspěch** Navrhuje výběr bodů zájmu na mapě, nenapadlo jej přidat ovlivněnou linku.	**Neúspěch** Nevyplnil polohu, žádá přiblíženou mapu.	**Úspěch** Žádá ale automat. vyplnění lokality a změnu *lajku* na lépe vypovídající hodnocení.	**Částečný úspěch** Nespokojenost s nutností klepnout na mapu, má tendenci dávat *dislajky*.
4.	**Částečný úspěch** Hledá pomocí odjezdů, bez rozmyslu vybírá první jedoucí spoj.	**Částečný úspěch** Tváří se nejistě a zmateně, nedokáže popsat proč. Jinak v pořádku.	**Netestováno** z důvodu opomenutí	**Úspěch** Navrhuje ale umístit zpoždění a mimořádnosti na přehled spojení.
5.	**Úspěch**	**Úspěch**	**Částečný úspěch** Není hned jasné.	**Úspěch** Navrhuje styl ±.
6.	**Úspěch** Chvíli hledá možnost hledání spojení z mé polohy, nevyužívá Nejbližší.	**Neúspěch** Nechápe skrývání „Přidat směr“, nepoužívá historii a označuje Můj autobus moc brzy.	**Úspěch** Připomínkuje, že je nutné při zobrazení mapy interagovat pro zobrazení časů odjezdu	**Úspěch** Chválí mapu přehlednou spojení.
7.	**Úspěch** přes notifikaci	**Úspěch** přes Můj cíl	**Úspěch** přes panel nástrojů	**Úspěch** přes Můj cíl
8.	**Úspěch** přes notifikaci	**Úspěch** přes panel nástrojů	**Úspěch** přes panel nástrojů	**Úspěch** přes panel nástrojů

5.1.4 Doplňující otázky

Každému z testerů jsem ještě položit 3 doplňující otázky:

1. Podle čeho se zobrazuje zpoždění autobusů?
2. Byli byste ochotni klepat na nástupní tlačítko při každé cestě autobusem?
3. Co jiného byste v aplikaci zlepšili, případně co je pro vás nejzásadnější problém?

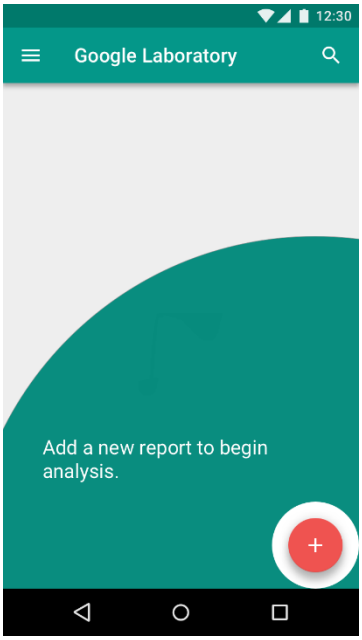
První otázka byla pouze ověřovací, zdali uživatel pochopil princip fungování aplikace. Z odpovědí, včetně těch na různé doplňující otázky, bylo zřejmé, že všichni zúčastnění testeři princip chápou.

Na druhou otázku byly rozporuplnější odpovědi. Vesměs šlo pochopit, že za aktuálního stavu to uživatelé dělat nebudou. Tři z nich (č. 2-4) by uvítali, kdyby bylo možné si nástup do autobusu označit (třeba 5 minut) předem. Z nich dva (č. 3 a 4) preferují způsob notifikace, kterou je nutné potvrdit. První tester by uvítal automatickou notifikaci bez předchozího vyhledání spoje nebo umístění widgetu do spouštěče aplikací.

V rámci třetí otázky navrhují první uživatel ukládání oblíbených linek a posouvání obrazovek v hlavní aktivitě swipe gestem. Za hlavní nedostatek považuje nepřehledný krátký přehled jízdního řádu. Ostatní uživatelé však tento přehled nevytýkali. Druhý tester ocenil celkový vzhled uživatelského rozhraní, ale zejména mu nepřijde praktické skrývání pole Kam na obrazovce Vyhledávání. Třetí a čtvrtý tester shodně vytýkali zejména nepraktičnost označování nástupu do autobusu. Ten čtvrtý navíc navrhuje zasílat notifikaci na výstup z autobusu předem, aby nezapomněl vystoupit.

5.1.5 Závěry testování

Testování ukázalo, že aplikace je funkční a uživatelé si jí cenní. Provedené úpravy oproti prototypu na obrazovce Vyhledávání měly pozitivní efekt na výsledky testů, avšak v této chvíli ještě není zcela srozumitelný rozdíl mezi vyhledáváním odjezdů a přímých spojení. Bude to dáno i tím, že do aplikace není implementována funkce vyhledávání spojení přestupních. Zásadnější problémy dělala orientace mezi výsledky vyhledávání při vybírání vhodného spoje v úkolu č. 4, to by mělo vyřešit zobrazování aktuálního zpoždění a piktogramů se symbolem druhu mimořádnosti v případě jejího výskytu přímo u jednotlivých spojů ve výsledcích.



Obrázek 28 – Material FAB onboarding podle Google (30)

Vytvoření onboardingu zlepšilo orientaci uživatelů, nicméně pouze jediný si pamatoval, že má při nástupu stisknout oranžové tlačítko. Navrhoval bych při prvním výskytu plovoucího tlačítka zobrazení další součásti onboardingu, která v souladu s Material designem upozorňuje na primární akci na obrazovce. (29)

6 ZÁVĚR

Práci považuji za úspěšnou. Podařilo se mi navrhnout a naimplementovat plně funkční mobilní aplikaci s cloudovým pozadím zobrazující jízdní řády autobusů a zaznamenávající jejich zpoždění a mimořádnosti v provozu.

Úvodní část jsem z velké části věnoval analýze požadavků cestujících, jelikož fakticky byla prováděna ještě před zadáním práce. Analýza stávajících řešení odhalila vysokou mezi požadavky cestujících a nedostatky existujících aplikací v podobě nedostatečných dat a informací o zpoždění a mimořádnostech. V rámci návrhu jsem představil uživatelské rozhraní, které dostalo život implementací na platformě Xamarin.Android. Tato implementace byla otestována čtyřmi vybranými uživateli, což potvrdilo funkčnost řešení a dalo podněty ke zlepšení do budoucna.

Pokud jde o to, co se zcela nepodařilo, jsou to funkce vyhledávání příjezdů nebo zobrazování mimořádností na obrazovkách s autobusem podle zadaného směru nebo u linek, které místem mimořádnosti projíždí, ale nahlašující uživatel je explicitně neoznačil. Odhad pozice a ujeté vzdálenosti na trase má také jeden skrytý problém v případě, že daná linka je polookružní – to znamená, že spoj jede část trasy po stejné cestě vícekrát. Tehdy je nutné například při odhadu sledovat směr pohybu, a to se v aktuální implementaci neděje. Kromě těchto věcí bych ještě zmínil nekompletní pokrytí softwarovými testy.

Toto vše se mi nepodařilo z důvodu nedostatku času, takže zmíněné funkce přesouvám mezi ty, které by měly být implementovány v budoucnu.

6.1 Budoucnost

Pro další rozvoj této práce se tedy nabízí mnoho příležitostí ke zlepšení. Některé jsem zmínil v předchozím odstavci, kromě nich je důležité zaměřit se na vychytání problémů zjištěných testováním. Zde je velký prostor zejména směrem k detekci nástupu do autobusu tak, aby uživatel nemusel s aplikací příliš interagovat, nebo k prevenci zneužívání a poškozování dat. Možnosti ke zlepšení jsou také na straně cloudové aplikace, ta potřebuje kvalitní databázové zázemí a dále by bylo dobré se zaměřit zejména na efektivitu algoritmů a snížení objemu přenášených dat přes internet. Trnem v oku je pak nedostatek kvalitních veřejných dat o jízdních řádech, který ale v této práci nelze vyřešit bez vstupu dalších stran – ať už dopravců, organizátorů dopravy nebo zákonodárců.

Jsem velmi rád, že jsem tuto práci mohl tvořit.

REFERENCE

1. **Český statistický úřad.** Přeprava věcí a osob, přepravní výkony. *Veřejná databáze*. [Online] 2017. [Citace: 9. 5. 2019.] https://vdb.czso.cz/. Kód DOP05-D/1.

2. **Správa železniční dopravní cesty, s.o.** Podíl jednotlivých dopravců na výkonech sítě SŽDC. [Online] 6. 5. 2019. [Citace: 14. 5. 2019.] https://www.szdc.cz/documents/50004227/50167315/podíl+výkonů+032019.pdf.

3. **StatCounter.** Mobile Operating System Market Share Czech Republic. *StatCounter Global Stats*. [Online] [Citace: 20. 5. 2019.] http://gs.statcounter.com/os-market-share/mobile/czech-republic.

4. **MAFRA, a. s.** Jízdní řády IDOS. *Aplikace na Google Play*. [Online] Google LLC, 9. 4. 2019. [Citace: 29. 4. 2019.] https://play.google.com/store/apps/details?id=cz.mafra.jizdnirady&hl=cs.

5. **Seznam.cz, a.s.** Jízdní řády. *Aplikace od Seznamu*. [Online] [Citace: 29. 4. 2019.] https://aplikace.seznam.cz/jizdnirady.html.

6. **Slížek, David.** Seznam: nasadili jsme data CHAPSu na Mapy.cz a koupili jsme Pubtran. *Lupa.cz*. [Online] 29. 4. 2015. [Citace: 6. 4. 2019.] https://www.lupa.cz/clanky/seznam-nasadili-jsme-data-chapsu-na-mapy-cz-a-koupili-jsme-pubtran/. ISSN 1213-0702.

7. —. Tohle je výsměch. Úplné uvolnění dat o jízdních řádech CHAPSem se zase nekoná. *Lupa.cz*. [Online] 1. 9. 2015. [Citace: 21. 5. 2019.] https://www.lupa.cz/clanky/tohle-je-vysmech-uplne-uvolneni-dat-o-jizdnich-radech-chapsem-se-zas-nekona/. ISSN 1213-0702.

8. **Circlegate.** CG Transit. *Circlegate*. [Online] [Citace: 29. 4. 2019.] https://www.circlegate.com/cs/cgt.

9. **CHAPS, spol. s r.o.** MHD Tabule. *Aplikace na Google Play*. [Online] Google LLC, 17. 5. 2016. [Citace: 29. 4. 2019.] https://play.google.com/store/apps/details?id=com.pitr.mhdtabule&hl=cs.

10. **Operátor ICT, a. s.** Aplikace PID Lítačka. [Online] [Citace: 29. 4. 2019.] https://app.pidlitacka.cz/.

11. **Google LLC.** *Material Design*. [Online] [Citace: 19. 4. 2019.] https://material.io/.

12. **Microsoft Corporation.** Android & iOS Apps with Xamarin. *.NET*. [Online] [Citace: 20. 5. 2019.] https://dotnet.microsoft.com/apps/xamarin/mobile-apps.

13. —. Shared Projects code sharing. *Xamarin Docs*. [Online] [Citace: 20. 5. 2019.] https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/shared-projects.

14. **Google LLC.** Distribution dashboard. *Google Developers*. [Online] 7. 5. 2019. [Citace: 20. 5. 2019.] https://developer.android.com/about/dashboards.

15. **Sůra, Jan.** Hotovo. Dcera Českých drah koupila za stovky milionů firmu Chaps. *Zdopravy.cz*. [Online] 24. 10. 2017. [Citace: 21. 5. 2019.] https://zdopravy.cz/hotovo-dcera-ceskych-drah-koupila-za-stovky-milionu-firmu-chaps-3423/. ISSN 2570-7868.

16. **České dráhy.** Výroční zpráva skupiny České dráhy 2018. 2019.

17. **Regionální organizátor Pražské integrované dopravy.** Otevřená data PID. *Pražská integrovaná doprava*. [Online] [Citace: 20. 5. 2019.] https://pid.cz/o-systemu/opendata/.

18. **Google LLC.** Static Transit Reference. *Google Developers*. [Online] [Citace: 20. 5. 2019.] https://developers.google.com/transit/gtfs/reference/.

19. **Microsoft Corporation.** Routing to controller actions in ASP.NET Core. .*NET Docs*. [Online] [Citace: 21. 5. 2019.] https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-2.2#attribute-routing.

20. **Google LLC.** Save key-value data. *Android Developers*. [Online] [Citace: 21. 5. 2019.] https://developer.android.com/training/data-storage/shared-preferences.

21. **Autor neznámý.** Haversine formula. *Rosetta Code*. [Online] 2. 12. 2011. [Citace: 18. 5. 2019.] http://rosettacode.org/wiki/Haversine_formula#C.23.

22. **Google LLC.** What is Instance ID? *Google Developers*. [Online] [Citace: 22. 5. 2019.] https://developers.google.com/instance-id/.

23. **Mew, Kyle.** *Android Design Patterns and Best Practice*. Birmingham : Packt Publishing, 2016. ISBN 978-1-78646-721-8.

24. **Microsoft Corporation.** Foreground Services. *Xamarin Docs*. [Online] [Citace: 22. 5. 2019.] https://docs.microsoft.com/en-us/xamarin/android/app-fundamentals/services/foreground-services.

25. **Werner, Tomáš.** Optimalizace. *Elektronická skripta předmětu B0B33OPT*. [Online] 29. 1. 2019. https://cw.fel.cvut.cz/b181/_media/courses/b33opt/opt.pdf.

26. **Suleiman.** Onboarding with Android ViewPager: The Google Way. *Grafix Artist*. [Online] 16. 12. 2015. [Citace: 9. 5. 2019.] https://blog.iamsuleiman.com/onboarding-android-viewpager-google-way/.

27. **Ombura, Martin Jr.** Android Design—Collapsing Toolbar: ScrollFlags Illustrated. *Medium*. [Online] 30. 1. 2018. [Citace: 3. 5. 2019.] https://medium.com/martinomburajr/android-design-collapsing-toolbar-scrollflags-e1d8a05dcb02.

28. **Osherove, Roy.** *The Art of Unit Testing*. Greenwich : Manning, 2009. ISBN 978-1-933988-27-6.

29. **Google LLC.** Onboarding. *Material Design*. [Online] [Citace: 23. 5. 2019.] https://material.io/design/communication/onboarding.html#quickstart-model.

30. —. *Nudge users to take the first key action*. Material Design, Onboarding : 2016.