

MATEMATICKO-FYZIKÁLNÍ FAKULTA

Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Filip Čižmář

Analýza real-time dat vozidel městské hromadné dopravy

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. Mgr. Martin Nečaský, Ph.D.

Studijní program: Informatika

Studijní obor: SW a datové inženýrství

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V dne
Podpis autora

Především děkuji svému vedoucímu, který mi pomohl najít zajímavé zaměření mé práce, umožnil přístup k otevřeným datům a pomohl při vypracování.

Stejně tak děkuji i Janu Vlasatému, který mi poskytl odbornou pomoc při získávání dat z datové platformy Golemio a inspiraci pro obsah mé práce.

Název práce: Analýza real-time dat vozidel městské hromadné dopravy

Autor: Filip Čižmář

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. Mgr. Martin Nečaský, Ph.D., Department of Software Engineering

Abstrakt: Tato práce se zaměřuje na analýzu dostupných otevřených real-time dat z vozidel hromadné dopravy v Praze a okolí. Jejím cílem je poskytnout základní statistické informace a na základě historických dat zlepšit odhad zpoždění spoje na trase mezi dvěma referenčními body. Jako vedlejší produkt vytvoří aplikaci pro webové rozhraní, kde zobrazí aktuální polohy spojů do mapového podkladu a rozšiřující infomace o nich. Aplikace bude aktivně interagovat s uživatelem.

Klíčová slova: zpoždění MHD otevřená data veřejná doprava

Title: Analysis of real-time data of public transport vehicles

Author: Filip Čižmář

Department: Department of Software Engineering

Supervisor: doc. Mgr. Martin Nečaský, Ph.D., Katedra softwarového inženýrství

Abstract: Abstract. Tato práce se zaměřuje na analýzu dostupných otevřených real-time dat z vozidel hromadné dopravy v Praze a okolí. Jejím cílem je poskytnout základní statistické informace a na základě historických dat zlepšit odhad zpoždění spoje na trase mezi dvěma referenčními body. Jako vedlejší produkt vytvoří aplikaci pro webové rozhraní, kde zobrazí aktuální polohy spojů do mapového podkladu a rozšiřující infomace o nich. Aplikace bude aktivně interagovat s uživatelem.

Keywords: delay open data public transport

Obsah

Úvod	3
1 Analýza	5
1.1 Úvod	5
1.2 Popis problému odhadu zpoždění	5
1.2.1 Příklad nelineárního profilu trasy	6
1.2.2 Současná řešení	8
1.3 Analýza zdroje dat	8
1.3.1 Přístup k datům	8
1.3.2 Analýza statických dat	13
1.4 Analýza vizualizačních nástrojů	14
1.4.1 Současná řešení	14
2 Návrh řešení	17
2.1 Úvod	17
2.1.1 Funkční a kvalitativní požadavky	17
2.2 Zpracování vstupních dat	19
2.2.1 Databáze	19
2.2.2 Plnění databáze	21
2.3 Algoritmus odhadu zpoždění	22
2.4 Vizualizace dat	22
2.4.1 Funkční požadavky	22
2.4.2 Poskytovatelé mapových podkladů	23
3 Implementace	25
4 Vizualizace dat	26
4.1 Klientská část	26
4.2 Serverová část	27
5 Algoritmus odhadu zpoždění	29
5.1 Základní předpoklady	29
5.1.1 Analýza dat	29
5.2 Návrh modelu	30
5.2.1 Lineární model	31
5.2.2 Polynomiální model	31
5.2.3 Model pomocí konkávního obalu	31
6 Porovnání se stávajícím řešením	34
7 Navrhy na zlepšení	35
8 Statistiky	36
Závěr	37
Seznam použité literatury	38

Seznam obrázků	39
Seznam tabulek	40
Seznam použitých zkratek	41
A Přílohy	43
A.1 První příloha	43

Úvod

Městská hromadná doprava v Praze a Středočeském kraji je jeden z hlavním pilířů přepravy osob na tomto území. Jejím rozsahem a důležitostí se přímo dotýká každého z nás a její fungování do značné míry ovlivňuje naše konání v krátkém i dlouhém časovém horizontu.

Každého cestujícího v přepravě jistě někdy trápilo zpoždění svého spoje. To člověka přivádí k myšlenkám jestli by nebylo možné určit s jakou pravidelností, pokud s nějakou, takové zpoždění vznikají. A jestli by nemohl být informován za včasu o vzniklé anomálii a vzniklému zpoždění.

Cílem této práce je zpřesnit odhad zpoždění vozidel VHD, zejména autobusů, na trase mezi dvěma sousedícími zastávkami. Dále pak tyto výsledky vizualizovat v mapových podkladech.

Ve vymezené oblasti operuje spousta soukromých i městských dopravců. Ti kteří spadají do naší zájmové oblasti zastřešuje organizace ROPID, která objednává jednotlivé spoje. Pro naši práci je důležité, že organizace ROPID zadala jednotlivým dopravcům vysílat aktuální polohy jejich vozů. Tato data o polohách jsou přes zprostředkovatele zveřejňována na pražské datové platformě zvané Golemio¹, jež je ve správě společnosti Operátor ICT, a. s., která je vlastněná hlavním městem Praha. Takových spojů, o kterých máme všechna požadovaná data je v pracovní den vypraveno necelých deset tisíc.²

V době návrhu práce, kvůli právním komplikacím a složitost informačního systému³, nebyly k dispozici real-time data z majoritního dopravce na území Prahy Dopravního podniku Prahy. Avšak protože je práce zaměřená na odhad zpoždění spoje mezi dvěma sousedícími zastávkami na trase, má tedy větší význam odhadovat zpoždění mezi zastávkami, mezi kterýma je větší vzdálenost. A to jsou převážně spoje jednoucí mimo Prahu. Proto tato data z DPP nenabývají takové důležitosti, jako data od dopravců operujících mimo Prahu. Vzhledem k tomu, že zbylí dopravci využívají převážně autobusy k přepravě cestujících, bude práce vypracována pouze s ohledem na autobusovou dopravu.

V práci se tedy pokusíme využít dostupná otevřená real-time data k získání infomarcí o zpoždění spojů na trase a využít je k lepším odhadům zpoždění. Řešení ovšem není pouze založeno na real-time datach, ale využívá také statická data o jízdních řádech nebo zastávkách hromadné dopravy, jejichž zdrojem je přímo ROPID⁴, a také mapové podklady. Ty jsou potřeba zejména pro vizualizaci zastávek a jízdních řádů nebo vykreslní tras spojů přímo do mapy. Avšak i tyto statická data jsou dostupná přímo z datové platformy pomocí stejného rozhraní jako data real-timová.

¹www.golemio.cz

²ze dne 20. 2. 2020 podle testovacích dat

³www.irozhlas.cz/zpravy-domov/data-o-poloze-vozidel-dpp-mhd-tramvaje-autobusy-praha-hrib-soud-informace_1904290600_kno

⁴pid.cz/o-systemu/opendata/

Protože disponujeme daty o aktuálních polohách vozidel MHD, která navíc rozšíříme o lepší odhady zpoždění. Nabízí se jejich využití tak, že budou vynezena do mapy a tím vznikne vizuálně přívětivé uživatelské prostředí pro prohlížení aktuálního stavu sítě vozidel. V práci tedy navrhujeme a implementujeme uživatelskou aplikaci, která vozidla zobrazí a bude komunikovat s uživatelem tak, že na žádost zobrazí více informací o daném spoji nebo vybrané zastávce.

1. Analýza

V této kapitole je detailně popsán problém a způsoby jeho navrženého a současného řešení. Dále zdroje dat se kterými budeme v této práci pracovat.

1.1 Úvod

Spoje které zajišťují hromadnou dopravu jezdí podle jízdních řádů, které definují jejich trasu. Trasa se udává sekvencí projíždících zastávek, časy příjezdu a odjezdu do, resp. z těchto zastávek a vzdáleností zastávek od výchozího bodu spoje. Tyto zastávky jsou zpravidla jediné referenční body u kterých je možno zjistit skutečné zpoždění, nebo předjetí (dále uvažováno jako zpoždění se zápornou hodnotou). Dále jsou součástí jízdních řádů také velice detailní nákresy tras každého spoje, formou lomené čáry definovanou posloupností souřadnic, kde každý bod je doplněn o jeho vzdálenost od výchozího bodu spoje.

Délka trasy mezi dvěma referenčními body nezřídku dosahuje i několika desítek kilometrů¹. Na těchto úsecích mohou vznikat mimořádné události, které se dají predikovat jen s těží. Nicméně ve většině případů je průběh jízdy ovlivněn pouze obvyklým provozem v dané denní době.

Detailní rozbor počtu průjezdů mezi zastávkami v daných vzdálenostech je vidět na grafu 1.1. Kde průjezdem se myslí každý jednotlivý průjezd vozidla mezi danou dvojicí zastávek v daný den. Data jsou platná pro spoje jedoucí v 20. 2. 2020.

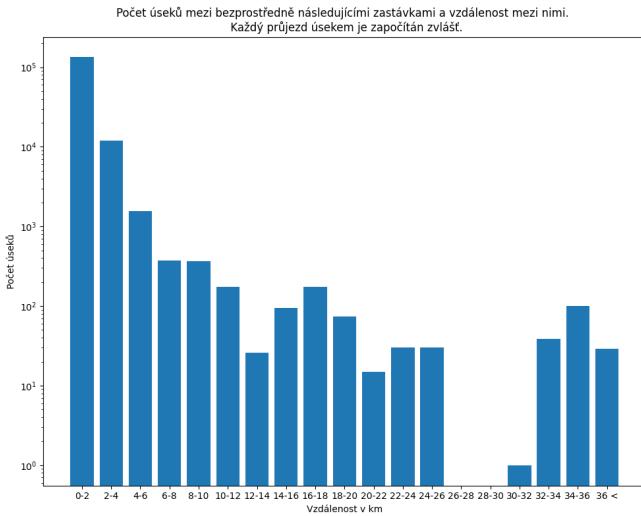
1.2 Popis problému odhadu zpoždění

Řešený problém se týká případu, kdy vozidlo projízdí mezi dvěma referenčními body a tato trasa má části, ve kterých vozidlo jede různou rychlostí. Např. vozidlo při vyjíždění z města jede mnohem pomaleji než při jízdě mezi městy. Takových úseků, na kterých se rychlosť jízdy liší může být na trase více a nedají se všechny jednoduše detektovat.

Tato Práce tedy modeluje profily jízd mezi referenčními body. A na základě toho zpřesnit odhad zpoždění. Tento odhad by měl být mnohem přesnější než současné odhady, které odpovídají tomu, že vozidlo jede konstantní rychlostí po celou dobu jízdy. Nebo je takové možné brát jako aktuální zpoždění spoje poslední změřené zpoždění při průjezdu nějakým referenčním bodem (zastávkou, nebo např. pro tramvaje se používají návěstidla).

Přidaná hodnota je tedy v tom, že Práce navrhne takové modely, které nebudou penezalizovat zvyšováním zpoždění za pomalou jízdu v úsecích, které se pomaleji projíždějí vždy. A také naopak zvýhodňovat snížením zpoždění za

¹Podle dat pro spoje jedoucí v 20. 2. 2020 je medián vzdáleností mezi zastávkama, mezi kterýma projede alespoň jeden spoj denně 943 m. Průjezdů mezi zastávkami ve vzdálenosti více než 10 kilometrů je 784, přičemž průjezdů mezi zastávkami ve vzdálenosti alespoň 2 km je přibližně 15000



Obrázek 1.1: Graf počtu úseků mezi následujícími zastávkami a vzdálenotí mezi nimi.

rychlou jízdu v úsecích, které se projízdějí rychle. Pokud bychom se tedy podívali na změny zpoždění na trase mezi dvěma referečními body, v ideálním případně by měli být nulové.

Pro řešení odhadu toho typu spoždění stačí navrhnout systém na odhat zpoždění v půběhu jízdy mezi referenčními body z historických dat jízd.

Pro vyloučení všech pochybností je hodno uvést, že se naše Práce nesnaží předpovědět zpoždění, které spoj může nabrat vzhledem k dosavadnímu průběhu trasy. Tedy např. nijak nezohledněuje to, že spoj právě stojí v mimořádné koloně a dalo by se tedy předpokládat, že zpoždění bude rychle růst i v následujících minutách. Ale naopak Práce se snaží odhadnou zpoždění v danném bodě na trase vzhledem k obvyklému profilu jízdy. Tedy např. pokud by výše uvažovaná kolona byla pravidelná Práce ji zohlední ve statistických modelech.

1.2.1 Příklad nelineárního profilu trasy

Celé ilustrováno na příkladě jízd mezi dvěma zastávkama K letišti a ZLičín, kde je nelineární profil jízdy vidět velice dobře. Jedná se totiž o trasu přesně odpovídající popisu problému.

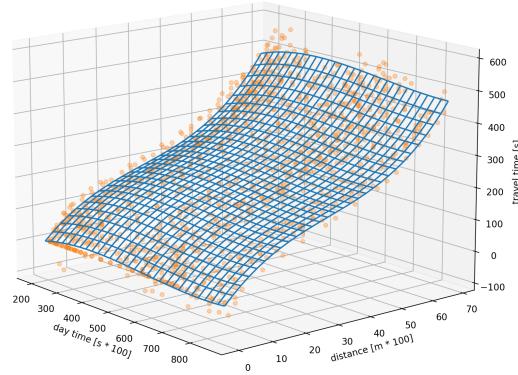
Popsané rozdíly v rychlosti a nelineární profil trasy je patrný na grafu 1.2. Za povšimnutí táké stojí viditelné spomalení průjezdů v ranní špičce, 7–9 hodina ráno.

Na obrázku 1.3 je pro bližší představu popsané trasy vidět trasa spoje vykreslená do mapy.

Rozbor trasy

Celá tato trasa má necelých 7 km a její průjezd spojem VHD trvá 10 minut. Prvních 600 metrů je vedeno po obecní komunikaci, přes křižovatku a nájezd na

K Letišti → Zličín



Obrázek 1.2: Modrá plocha značí vymodelovaný profil trasy. Oražové body jsou jednotlivé vzorky poloh vozidel. Data pro graf jsou ze dnů 20.–21. 2. 2020



Obrázek 1.3: Trasa mezi zastávkama K Letišti a Zličín. Zdroj: mapy.cz

Pražský okruh. Průměrná rychlosť vozidel byla 35 km/h^2 .

Dále trasa pokračuje přes Pražský okruh rovně až do vzdálenosti 4.9 km od zastávky K Letišti, kde začíná nájezd na ulici Na Radost. Dá se předpokládat, že vozidla se na komunikaci vyšší třídy pohybují rychleji což dokazuje, že na tomto úseku trasy se průměrná rychlosť vozidel zvýšila na 63 km/h .

Poslední úsek se tedy skládá z výjezdu z Pražského okruhu, průjezdu křižovatkou, jízdy po obecní komunikaci a vjezdu do stanice Zličín. Délka úseku je 2 km. Průměrná rychlosť za celou trasu se na tomto úseku snížila na 55 km/h .

1.2.2 Současná řešení

Algoritmus na odhad aktuálního zpoždění mezi dvěma referenčními body již existuje a je součástí Datové Platformy – Golemio, ze kterého se čerpají data pro tuto práci. (Detailní popis dat uveden v kapitole ??.)

Nicméně tento algoritmus nijak nezohledňuje variabilitu profilu trasy. Totiž v tomto řešení je nahlíženo na postup vozidla na trase jako na lineární funkci vůči času. Je ovšem zřejmé, že rychlosť vozidel není konstantní, neboli doba jízdy není linárně závislá na ujeté vzdálenosti.

Proto je potřeba tento odhad zpřesnit, což je cílem naší práce. K tomuto cíli jsme byli nasměrování v rámci schůzky s pracovníky společnosti OICT, kde bylo řečeno, že toto je problém současného řešení, který je potřeba vyřešit.

1.3 Analýza zdroje dat

V této kapitole je popsán zdroj real-timových dat o polohách vozidel využívané v této práci.

1.3.1 Přístup k datům

Vozidla vysílají data o své poleze při různých událostech. Zejména pak při brzdění, rozjezdu, vyhlášení zastávky, nebo jinak každých 20 sekund³.

Taková data pak přímo putují k provozovateli systému na monitorování vozidel, kterým je společnost Kapsch jakožto partner DPP. Ten však tato data zpracovává a posílá ke zveřejnění na platformě Golemio. Bohužel při tomto procesu zpracování se vytratí informace o události v jaké byly data pořízeny. Tedy informace o příjezdu nebo odjezdu ze zastávky jsou zjistitelné pouze z GPS souřadnic a následném odhadu pozice vozidla na trase dané linky.

²Počítáno podle vozidel, které poslaly polohu v 600m (resp. 4.9km, resp. 6.6km pro další údaje o rychlosti) vzdálenosti od zastávky. Počet záznamů o poloze vozidel se v různých vzdálenostech liší.

³Řečeno zaměstnancem OICT na schůzce 4. 5. 2019

Po té co jsou tyto data přeneseny do společnosti Operátor ICT by měla být zveřejněna, nicméně data ve výše popsané podobě jsou poměrně chudá, proto je k nim přidáno více atributů. Jedná se o dopočet poslední projeté zastávky, ujeté vzdálenosti od výchozí stanice, zpoždění v poslední zastávce.

Z pohledu této práce je nejzajímavější informace o vzdálenosti, kterou vozidlo urazilo od jeho výchozí zastávky. Dále jsou přidána data o jízdních rádech a zastávkách jejichž původcem je ROPID.

Real-time data o polohách, která jsou již neplatná (zastaralá) se neposílájí (posílá vždy pouze neaktuální informace) a i z Datové platformy jsou data po pár minutách nenávratně smazány.

Dokumentace

Na úvod je nutné poznamenat, že datová platforma je stále ve vývoji a formát dat se může měnit. S tím mohou přicházet určité výpadky a problémy. K jednomu takovému výpadku došlo i při vývoji této práce, kdy po dobu 14 dnů plafomarma vůbec neodpovídala na dotazy nebo vracela prázdné datasety.

Současně s využívanou verzí API, je nasazená i pokročilejší API ve verzi 2, které obsahuje více informací a je přehledněji upravena. Nicméně při zahájení vývoje této práce nebyla verze 2 k dispozici, proto jsou využívána data pouze ze starší verze.

Oficiální dokumentace datové platformy⁴ je poměrně zastaralá sama o sobě tak, že aktuální sada parametrů jí neodpovídá a neobsahuje žádné popisy nebo vysvětlení dat. Proto vysvětlení jednotlivých atributů se zakládá na intuitivním pochopení nebo vyplynulo z jednání se správci platformy. V následujících kapitolách bude popsán formát dat, tak jak přichází ze zdroje. Ten se může od oficiálně vystavené dokumentace lišit. A také budou popsány pouze atributy využívané v této práci nebo zajímavé pro její budoucí rozvoj.

Každá datová sada je exportována ve formátu GEOJSON pokud se jedná o geografická data, nebo jinak ve formátu JSON. Přistupuje se k nim přes jednotné webové rozhraní pomocí HTTP požadavku daného URL adresou a jeho hlavičkou.

TODO reference na specifikace geojson??? detailnejsi popis pristupoveho api???

<https://tools.ietf.org/html/rfc7946>

Ačkoli se dokumentace tváří tak, že data jsou exportována ve formátech JSON nebo GEOJSON, většinou formát dat není přesně podle specifikace těchto formátů. Například může být uveden atribut `wheelchair_accessible`, který je typu `bool` a je nastaven na hodnotu `True`, nicméně podle specifikace se tyto hodnoty píší s malým písmenem. Pro tuto práci to sice nepředstavuje komplikaci, protože tento atribut není potřeba, ale mohlo by se stát, že některé parsery JSONu vyhodnotí řetězec jako nevalidní a skončí chybou.

TODO rozepsat: Celá datová platforma Golemio je pojatá jako Open Source projekt.

⁴Golemio: <https://golemioapi.docs.apairy.io>

Pozice vozidel

Ze zveřejněných dat na této platformě jsou nejdůležitější data pro tuto práci pozice vozidel. Jelikož se jedná o real-time data, data rychle zastarávají a je nutné je velmi často aktualizaovat.

Využívané atributy jsou:

- coordinates aktuální GPS souřadnice vozidla
- origin_timestamp čas zachycení pozice vozidla, v časovém pásmu UTC
- gtfs_trip_id unikátní identifikátor tripu pro spárování s jízdním řádem
- gtfs_shape_dist_traveled vzdálenost vozidla uražená od začátku tripu v metrech
- delay_stop_departure pozdění zachycené při odjezdu z poslední projeté zastávky v sekundách

Příklad dat popisující aktuální polohu vozidla, na kterém je možno vidět strukturu dat i další atrubuty. Řada z nich je pro tuto práci zbytečná. Dále je možno si povšimnout atrubutu all_positions, který obsahuje všechny zaznamenané pozice daného vozidla na jeho aktuální trase, tento atribut je z důvodů objemu dat volitelný a pro tuto práci se nevyužívá.

```
"geometry":{  
    "coordinates":[14.91724,50.41881],  
    "type":"Point"  
},  
"properties":{  
    "trip":{  
        "cis_agency_name":"ČSAD Česká Lípa",  
        "cis_id":"260467",  
        "cis_number":3008,  
        "cis_order":2,  
        "cis_parent_route_name":"467",  
        "cis_real_agency_name":"ČSAD Česká Lípa",  
        "cis_short_name":null,  
        "cis_vehicle_registration_number":1073,  
        "gtfs_route_id":"L467",  
        "gtfs_route_short_name":"467",  
        "gtfs_trip_id":"467_252_200105",  
        "id":"2020-02-23T18:50:00Z_260467_467_3008",  
        "start_cis_stop_id":30107,  
        "start_cis_stop_platform_code":"A",  
        "start_time":"19:50:00",  
        "start_timestamp":"2020-02-23T18:50:00.000Z",  
        "vehicle_type":4,  
        "wheelchair_accessible":true  
},
```

```

"last_position": {
    "bearing": 20,
    "cis_last_stop_id": 21393,
    "cis_last_stop_sequence": 28,
    "delay": 261,
    "delay_stop_arrival": null,
    "delay_stop_departure": 287,
    "gtfs_last_stop_id": "U3389Z1",
    "gtfs_last_stop_sequence": 30,
    "gtfs_next_stop_id": "U2987Z30",
    "gtfs_next_stop_sequence": 31,
    "gtfs_shape_dist_traveled": "64.1",
    "is_canceled": false,
    "lat": "50.41881",
    "lng": "14.91724",
    "origin_time": "21:29:37",
    "origin_timestamp": "2020-02-23T20:29:37.000Z",
    "speed": 20,
    "tracking": 2,
    "trips_id": "2020-02-23T18:50:00Z_260467_467_3008"
},
"all_positions": {
    "features": [],
    "type": "FeatureCollection"
}
},
"type": "Feature"

```

Jízdy

Dále jsou k dispozici data o každé jízdě. To je popis trasy vozidla, včetně zastávek a časů příjezdů a odjezdů do/z nich. Také může být vyžádáno k informacím o jízdě připojit celý shape trasy, tj. lomená čára kopírující celou trasu daného tripu po povrchu Země.

Míra unikátnosti identifikátorů těchto tripů je předmětem dohadů a zřejmě jsou pod správou plánovačů MHD, nicméně pro účely této práce může být předpokládáno, že každá jízda má vlastní jízdní řád, který se váže na čas a každá jízda jede nejvýše jednou za den.

- trip_headsignnápis na čele vozidla, typicky cílová stanice
- route_idčíslo linky
- trip_idunikátní identifikátor tripu pro spárování s real-time daty, pravděpodobně odpovídá atributu **gtfs_trip_id**

Navíc s každým tripem může být vyžádáno zaslání seznamu zastávek, kterýma projízdí. Zde jsou k dispozici informace vázající se k danému průjezdu zastávkou. Každá uvedená zastávka s sebou nese i kompletní informaci o sobě, tedy má stejnou informační hodnotu jako samostatný dotaz na jednotlivé zastávky z datové sady zastávek.

Zastávky

- arrival_timečas příjezdu spoje do zastávky
- departure_timečas odjezdu spoje do zastávky
- shape_dist_traveled vzdálenost zastávky na trase od výchozího bodu daného tripu v metrech
- stop_id unikátní identifikátor zastávky
- coordinatesGPS souřadnice zastávky, často null, je třeba využít atributy stop\lat a stop\lon
- stop_name námenázev zastávky

Příklad dat popisujících jednu jízdu včetně zastávek. Seznam zastávek a body tras je zkrácen vzhledem k objemu dat.

```
"bikes_allowed":2,  
"block_id":"",
"direction_id":1,  
"exceptional":1,  
"route_id":"L421",
"service_id":"1111100-1",
"shape_id":"L421V4",
"trip_headsign":"Kolín, Nádraží",
"trip_id":"421_225_191114",
"trip_operation_type":1,
"trip_short_name":"",
"wheelchair_accessible":2,  
"stop_times": [  
    {"arrival_time":"14:14:00",  
     "arrival_time_seconds":null,  
     "departure_time":"14:14:00",  
     "departure_time_seconds":null,  
     "drop_off_type":"0",  
     "pickup_type":"0",  
     "shape_dist_traveled":0,  
     "stop_headsign": "",  
     "stop_id": "U2033Z5",  
     "stop_sequence":1,  
     "timepoint":null,  
     "trip_id": "421_225_191114",  
     "stop": {  
         "geometry": {  
             "coordinates": [  
                 null,  
                 null  
             ],  
             "type": "Point"  
         },  
         "properties": {
```

```

    "level_id":"",
    "location_type":0,
    "parent_station":"",
    "platform_code":"5",
    "stop_code":null,
    "stop_desc":null,
    "stop_id":"U2033Z5",
    "stop_lat":49.87486,
    "stop_lon":14.9078,
    "stop_name":"S\u00e1zava,Aut.st.",
    "stop_timezone":null,
    "stop_url":"",
    "wheelchair_boarding":0,
    "zone_id":"5"
  },
  "type":"Feature"
},
...
],
},
"shapes": [
  {
    "geometry":{
      "coordinates":[
        14.90778,
        49.87494
      ],
      "type":"Point"
    },
    "properties":{
      "shape_dist_traveled":0,
      "shape_id":"L421V4",
      "shape_pt_lat":49.87494,
      "shape_pt_lon":14.90778,
      "shape_pt_sequence":1
    },
    "type":"Feature"
  },
  ...
]

```

1.3.2 Analýza statických dat

Sběr dat probíhal ve dnech 20.–24. 2. 2020.

Data byla přebírána pouze z dat o každé jednotlivé jízdě a aktuálních poloh vozidel stahovaných každých 20 sekund. Tedy pokud určitou zastávkou žádný spoj neprojel nebo se uložení spoje z důvodu neúplných dat nezdařilo, může nějaká zastávka v databázi chybět. Stejně tak mohou chybět i nějaké spoje. Nejčastěji chybějící požadovaná data jsou informace o spoždění spoje v poslední zastávce, zcela nenalezený spoj ve zdroji dat a tedy chybějící jízdní řád. Nicméně jedná se zanedbatelné množství. Práce totiž nemůže počítat s poškozenými daty a není

jejím úkolem držet všechny zastávky, ale pouze ty kterými nějaký spoj projízdí. To z důvodu přehlednosti zastávek při vizualizaci, kde ukazovat nepoužívané zastávky nemá smysl a také pro odhadu zpoždění nedává smysl počítat něco pro zastávku, která není obsluhovaná.

Zastávky

Do databáze bylo celkem vloženo 5820 nástupišť⁵, které náleží celkem 2961 zastávkám. Ale naprostá většina (74 %) zastávek jsou párová, tedy mají pouze 2 nástupiště. Jednosměrných zastávek je 18 %, zastávek se 3 nástupišti jsou 3 %. Nejvíce nástupišť mají stanice Slaný Aut. nádr. (14), Černý Most (12), Kladno Autobusové nádraží (11).

Jízdy

Celkem bylo nalezeno 12788 spojů, z nich naprostá většina vyjela opakováně v následující den⁵.

Další rozbor dat na grafu 1.1 a v kapitole ??.

1.4 Analýza vizualizačních nástrojů

Jak bylo řečeno v úvodu, součástí práce je i vizualizace spočítaných dat.

To bude provedeno formou front endové aplikace, která zobrazuje mapu a do ní zanáší data o vozidlech VHD. Funkční požadavky této aplikace jsou inspirovány již existujícími řešeními tohoto problému.

1.4.1 Současná řešení

Vizualizaci vozidel VHD do mapy již nabízí několik portálů. Všechny jsou však poměrně strohé.

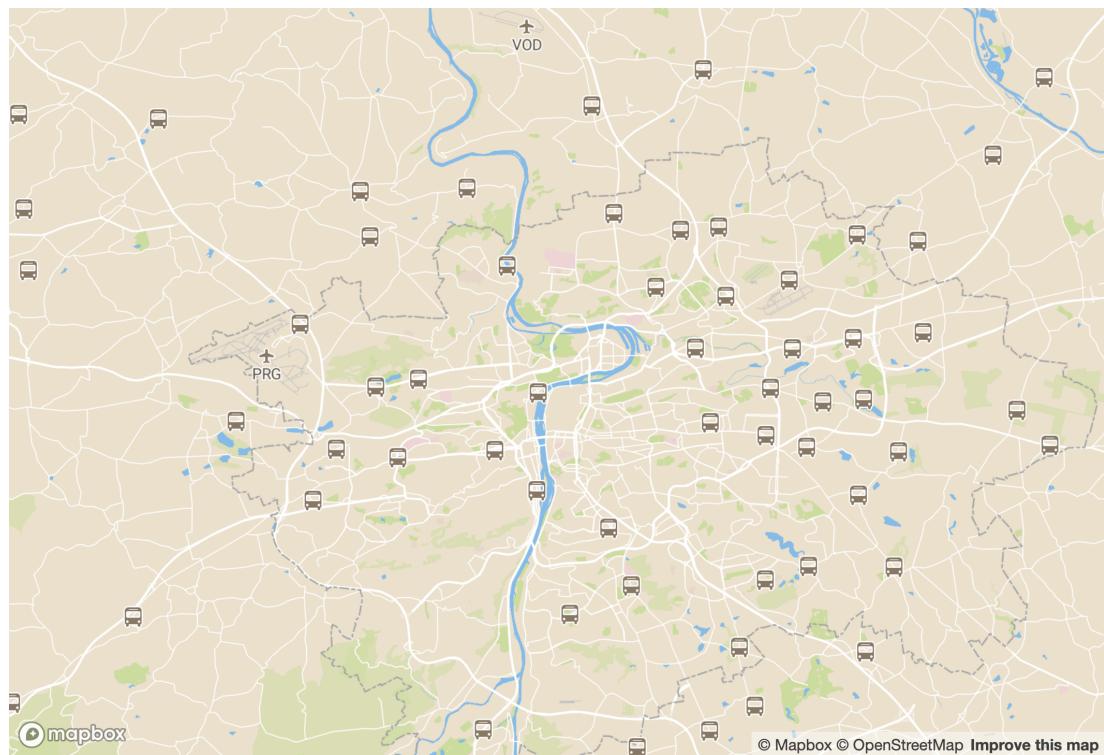
Golemio

Takovou mapu zobrazuje i samotný provozovatel datové platformy. Nicméně nejsou zde vidět ani čísla linek zobrazených autobusů, natož pak nějaké další informace.

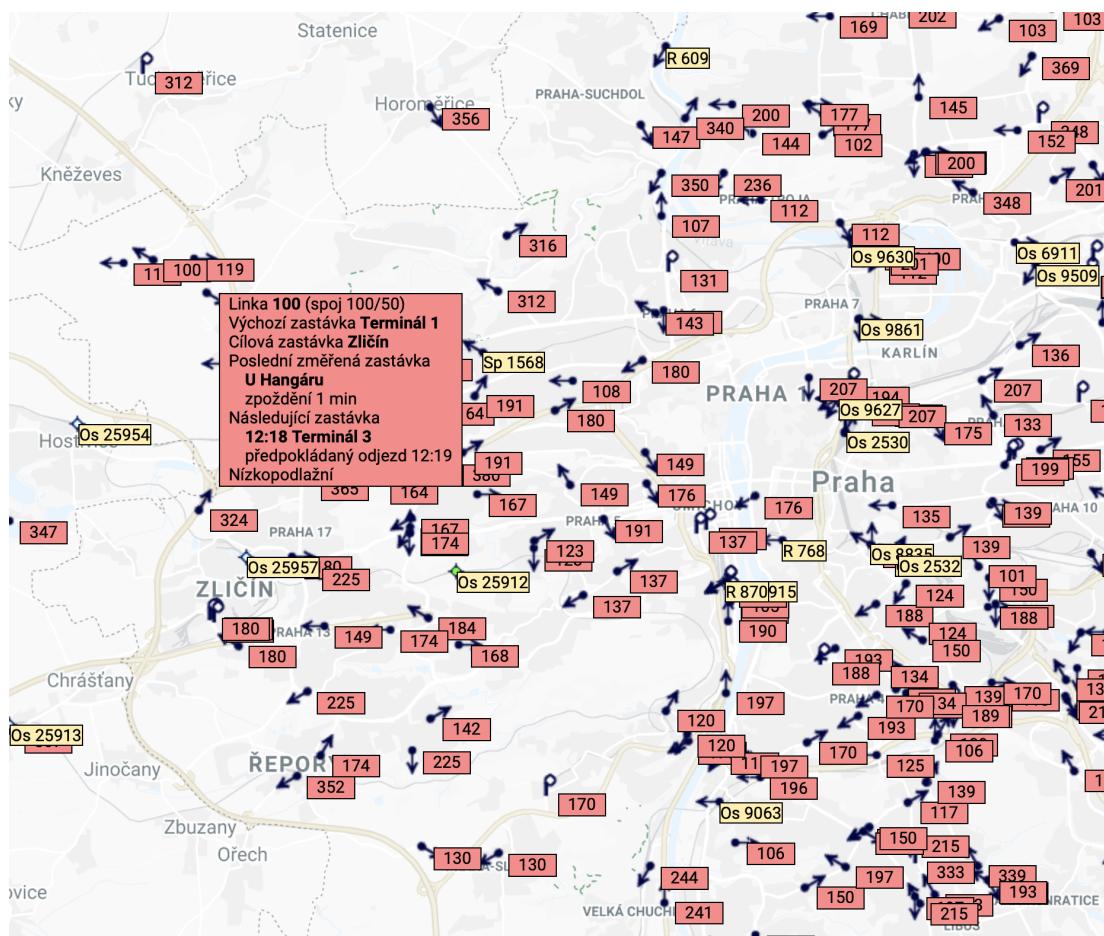
Tram-bus

Dalším poskytovatelem je portál tram-bus, který si vede o něco lépe. Ukazuje směr jízdy vozidel, čísla linek a po kliknutí informace o zpoždění a nejbližší zastávky. Pozn.: na mapě již jsou vidět spoje DPP, protože v době psaní této práce již byly data veřejné.

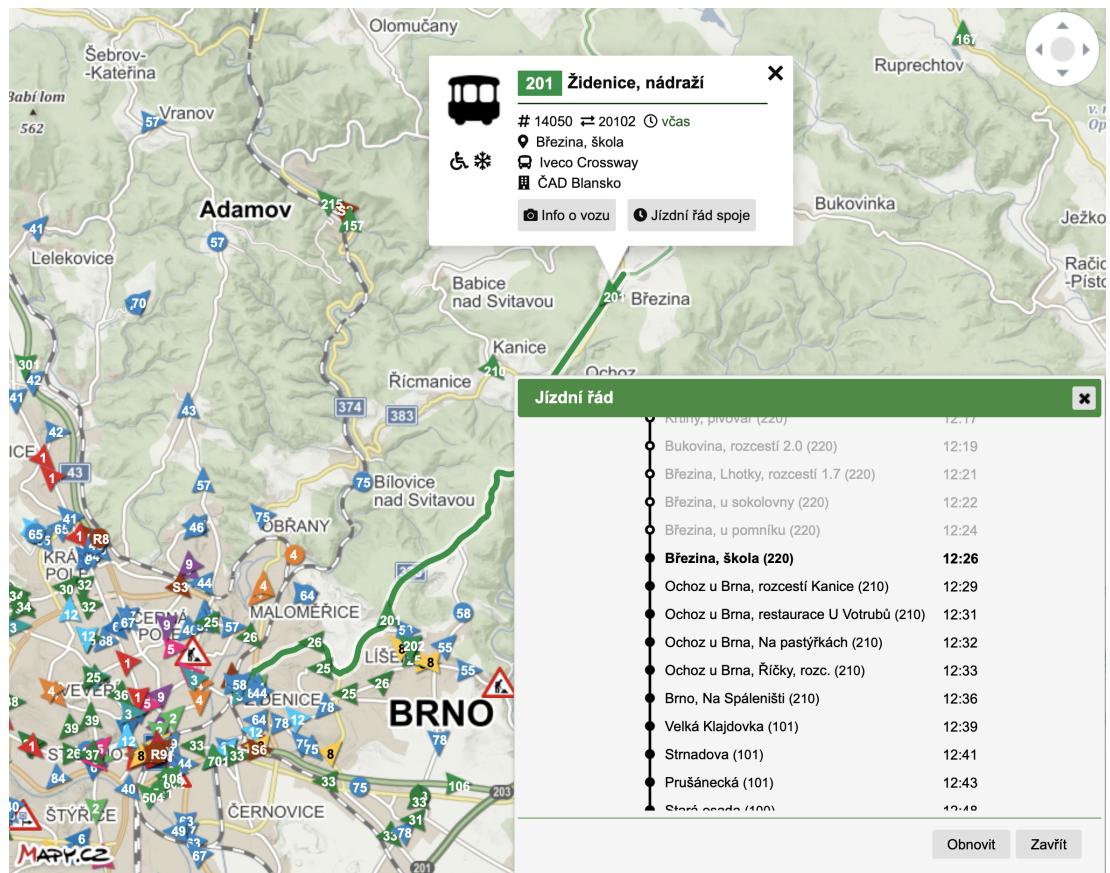
⁵Ze všech vypravených jízd ve dnech 20. 2. 2020 (9334) a 21. 2. 2020 (9428) jich 9051 bylo označeno ve zdrojových datech stejným identifikátorem, tedy z našeho pohledu sdílely jízdní řád a cestu.



Obrázek 1.4: Mapa z golemio.cz.



Obrázek 1.5: Mapa z www.tram-bus.cz.



Obrázek 1.6: Mapa z mapa.idsjmck.cz.

IDSJMK

Mimo Prahu je velice pěkně udělaná aplikace pro zobrazení vozidel IDSJMK (Integrovaný dopravní systém Jihomoravského kraje). Ten ihned po načtení stránky zobrazuje všechny dobravní prostředky, tedy tramvaje, autobusy a vlaky vše s čísly linek. Dále pak umožňuje po kliknutí na vybraný spoj zobrazit více informací včetně jízdního řádu.

Tato aplikace je po vizuální i funkční stránce dobrou inspirací pro tvorbu aplikace v této práci.

2. Návrh řešení

V této kapitole je popsán návrh technického řešení uvedených problémů.

2.1 Úvod

Běh celé aplikace bude rozdělen do dvou částí.

- Stahování a ukládání real-time dat o polohách vozidel do datového skladu, které budou doplněny o odhad zpoždění pro okamžité zveřejnění v uživatelské aplikaci.
- Modelování profilů jízd jednotlivých úseků. Tyto modely budou pak dále sloužit k odhadování zpoždění v budoucnu. Výpočet modelů bude prováděn jednou za delší časový úsek (nejlépe jednou za den).

Protože obě části jsou na sobě závislé v iniciálním běhu bude prováděna první část sběru dat bez odhadu zpoždění, nebo pomocí již existujícího triviálního lineárního odhadu.

Schéma návrhu celé aplikace a komunikační mapa jednotlivých komponent je ilusrrována na diagramu 2.1.

2.1.1 Funkční a kvalitativní požadavky

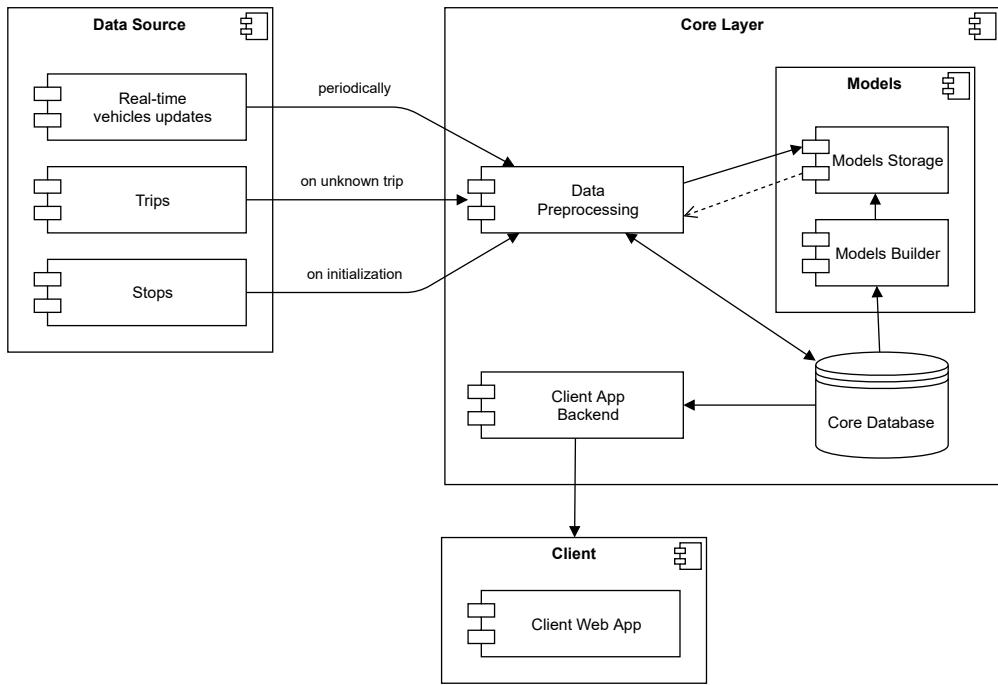
Nejprve specifikujme požadavky systému, na kterém se pak bude zakládat konkrétní návrh řešení backendu celé aplikace (bez vizualizace).

Funkční požadavky

- Popsaný odhad změny zpoždění na trase mezi dvěma referenčními body je nutné počítat v co nejkratším čase tak, aby cestující byli dobře informováni o stavu jejich spoje a mohli tyto informace využít např. při dobíhání spoje. A proto je potřeba zpracovávat data okamžitě po jejich vydání, spočítat odhad zpoždění a vystavit tato data veřejně. Vzhledem k tomu, že tato data velmi rychle zastarávají je nutné provádět tento proces co možná nejrychleji¹.
- Data o polohách vozidel VHD v Datové platfomě jsou aktualizována nejpozději každých 20 sekund, více v kapitole ???. Tedy pro minimalizaci rychlosti zastarávání dat a získání všech existujících vzorků dat o polohách je nutné data stahovat alespon každých 20 sekund.
- Odhad zpoždění se bude provádět na základě historických dat z posledních vyšších jednotek dnů². Tím se sníží dopad mimořádné události na předpovědní model, která může na trase vzniknout. Zároveň by však neměla být

¹Průměrná doba jízdy spoje mezi zastávkami je cca 5 min. Rozložení počtu úseků mezi zastávkami k délce jízdy mezi nimi je závislé a podobné rozložení vůči vzdálenosti ilustrované na grafu1.1.

²Pro demonstrativní účely této práce jsou využívány historická data pouze ze 4 dnů (2 pracovní a 2 víkendové).



Obrázek 2.1: UML diagram návrhu aplikace

započítávána data starší několik týdnů, protože dopravní situace se mění v závislosti na ročním období nebo také pokud je na trase delší omezení dopravy je požadováno, aby se takové omezení projevilo co možná nejdříve. Navíc se bude rozlišovat mezi daty z pracovních dnů a nepracovních dnů, to protože samotné jízdní řády se mohou lišit (doba jízdy mezi mezastávkama) a také se do velké míry liší hustota dopravy, která má velký vliv na profil jízdy. TODO do navrhů na zlepšení: Pro zpřesnění výsledků by bylo lepsi respektovat svatky, kazdy den v tydnu zvlášť atp.

- Zpracování historických dat bude probíhat vždy po delší době, nejlépe jednou za den. To umožní provádět náročnější výpočty, které by za normálního provozu neúměrně přetížily systém. Navíc vzhledem k povaze cíle práce ani není žádoucí zpracování historických dat provádět častěji než jednou denně, protože se nepokoušíme okamžitě reagovat na změnu dopravní situace.
- Uložená historická data budou strukturovaná tak, aby nad nimi šly provádět statistické výpočty minimálně o frekvencích spojů, vzdálenostech tras, zpoždění spojů.

Kvalitativní požadavky

- Řešení bude schopno při jedné aktualizaci zpracovat alespon³ 1000^3 vzorků poloh vozidel, kde 10 % vzorků může být o dosut neznámých jízdách. V tomto případě je potřeba stáhnout jízdní řád konkrétní jízdy a její jízdní profil, což navíc dotaz na zdroj dat.

³20. 2. 2020 mezi 7:00 a 7:10 bylo na trase přes 600 vozidel

- Vypočítané modely profilů jízd budou dávat odhad zpoždění lepší, než je lineární odhad. To znamená, že zpoždění vypočítaná pro každý přijatý vzorek polohy vozidla mezi dvěma referenčními body na trase bude mít menší rozptyl než lineární odhad zpoždění.

2.2 Zpracování vstupních dat

Struktura uložení dat se zakládá na struktuře zdrojových dat popsaných v kapitole 1.3 Analýza zdroje dat.

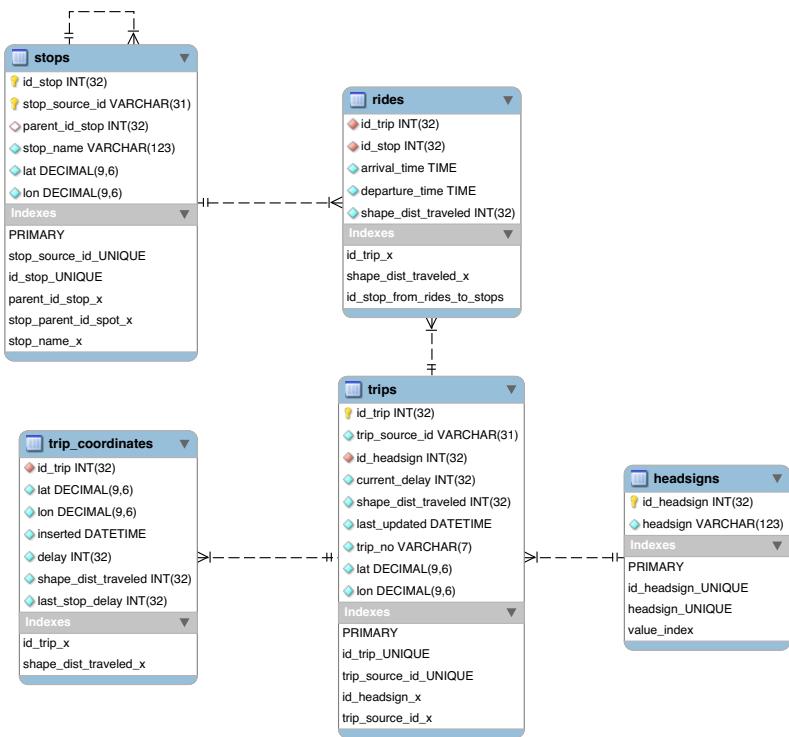
Na datové platformě jsou real-time data o vozidlech dostupná do historie řádově jednotek minut, což je naprosto nedostatečné pro jakékoliv pozdější využití v rámci této práce. Především pro počítání statistik a modelování profilů jízd nad daty je potřeba zřídit lokální databázi, která bude držet historická data tak, jak byla obdržena od zdroje. Navíc data jsou poskytována ve formátu JSON, který svou povahou není zrovna úsporný co se do velikosti souboru týče. Proto je vhodné zvolit ukládání dat v jiném formátu.

2.2.1 Databáze

Za tímto účelem tato práce využívá relační databázi obsluhovanou dotazovacím jazykem SQL. Struktura databáze je vyobrazena na EER diagramu 2.2⁴. Tato databáze se skládá z 5 tabulek. Jsou jimi:

- **trips** všechny objevené jízdy
 - **id_trip** unikátní identifikátor používaný v databázi
 - **trip_source_id** identifikátor tripu převzatý ze zdroje dat
 - **id_headsign** identifikátor nápisu pro daný trip
 - **current_delay** aktuální zpoždění tripu
 - **shape_dist_traveled** aktuální vzdálenost ujetá od výchozí stanice
 - **last_updated** čas poslední aktualizace, převzatý ze zdroje dat
 - **trip_no** číslo dané linky
- **headsigns** náписy nad vozidlem, cílová stanice
 - **id_headsign** unikátní identifikátor nápisu
 - **headsign** text nápisu
- **trip_coordinates** všechna historická real-time data
 - **id_trip** identifikátor tripu, ke kterému se záznam váže
 - **lat** zeměpisná šířka polohy vozidla

⁴SQL dotazy na sestavení celé databaze jsou definovány v příloze database.sql. Pro testovací, debugovací a demonstrační účely slouží navíc i jiné databáze, které jsou strukturou totožné jako produkční databáze.



Obrázek 2.2: EER diagram databáze.

- `lon` zeměpisná délka polohy vozidla
 - `inserted` čas vložení záznamu
 - `delay` zpoždění zachycené v poslední projeté stanici před pořízením záznamu
 - `shape_dist_traveled` vzdálenost ujetá od výchozí stanice tripu
- `stops` všechny zastávky
 - `id_stop` unikátní identifikátor zastávky
 - `trip_source_id` identifikátor zastávky převzatý ze zdroje dat
 - `parent_id_stop` identifikátor rodičovské zastávky, pokud existuje
 - `stop_name` název zastávky
 - `lat` zeměpisná šířka polohy zastávky
 - `lon` zeměpisná délka polohy zastávky
- `rides` trasa každého tripu, seznam zastávek s časy odjezdů a příjezd tvořící jízdní řád
 - `id_stop` identifikátor tripu
 - `id_stop` identifikátor zastávky
 - `arrival_time` čas příjezdu tripu do zastávky
 - `departure_time` čas odjezdu tripu ze zastávky
 - `shape_dist_traveled` vzdálenost zastávky od výchozí zastávky tripu

Atributy se jménem `*source_id` jsou pravděpodobně unikátní identifikátor entity ve zdroji dat, nicméně z dokumentace zdroje to nevyplývá. Také je tento identifikátor ukládán jako textový řetězec, ačkoli je tvořen pouze číslicemi a podtržítky, není nikde zaručeno, že jej lze jednoduše převést na číselný ko'd. Takže pro lepší výkon databáze je použito automaticky generované id typu INT.

Každá tabulka má několik indexů, které zlepšují výkon databáze při vkládání a hledání dat. Obzvláště pokud je atribut označen jako unikátní, kde se při každém vložení ověřuje unikátnost.

Databáza je nastavená tak, aby umožňovala získat všechny potřebné informace o vozidlech, ale hlavně přístup k historickým real-time datům a to separovaně pro dvojci referenčních bodů.

2.2.2 Plnění databáze

Tato databáze bude plněna skriptem, jehož bude naprogramován taky, aby vždy stál aktuální obraz dopravní situace a tato data uložil do dotabáze. Toto stahování z datové platformy probíhá podle následujícího algoritmu.

Algoritmus:

```
načti všechny dostupné zastávky
dokud skrip běží
    načti aktuální polohy vozidel
    pro každé nalezené vozidlo
    pokud jízda vozidla je známá
        aktualizuj data o jízdě
    jinak
        stáhni informace o jízdě
        zpracuj a vlož jízdu do databáze
```

Protože všechny infomace ukládané do databáze jsou důležité pro hlavní cíl této práce, tak pokud se vyskytne jízda, který neobsahuje některou z požadovaných infomarcí je pak automaticky zahozena. To je řešeno pomocí databázových transakcí tak, aby stav databáze byl vždy konzistentní. Transakce v obecném smyslu fungují tak, že můžeme měnit data v databázi (i více záznamů) a tyto změny se zapíší do samotné databáze až po potvrzení, že všechny změny byly provedeny správně, pokud během provádění změn nastane chyba, můžeme v jakékoli fázi provádění změn všechny dosud proveedené změny zahodit a vrátit se do původního stavu databáze před započetím transakce. Tedy pokud nejsou poskytnuta data ve formátu, který skript akceptuje, nebo nějaké povinné atributy chybí. Vložení celé jízdy nebude provedeno.

Nejčastěji chybějící atribut je zpoždění v poslední zastávce, toto je nutné vědět pro počítání zpoždění mezi refenrečními body (zastávkami). Absence této informace může být způsobena tím, že vozidlo vůbec nevysílá data potřebná k jejím dopočtení, pak nemá smysl jej do databáze zahrnovat. Nebo vozidlo už vysílá, ale ještě nezahájilo jízdu, tedy nemá žádnou poslední projetou zastávku, v takovém případě budou data ignorována až do doby příchodu první relevantní informace.

Mimo popsanou databázi se do určeného adresáře ukládají trasy jednotlivých jízd, která jsou ve fromátu GEOJSON jako lomená čára definována souřadnicemi. Navíc data o trasách jsou používána pouze pro vizualizaci a jsou přijímány vizualizačním nástrojem ve formátu GEOJSON, tedy tyto data není nutné vůbec transformovat a není nutné je držet v hlavní databázi.

Stejně tak i aktuální polohy vozidel jsou mimo databázi zapisovány do souboru, který je určen a formátován pro čtení webovou aplikací. Aktualizace tohoto souboru je provedena přednostně, ihned po načtení real-timových dat. Tím se zabrání nechtěnému čekání na aktualizaci celé databáze, která může trvat jednotky sekund.

2.3 Algoritmus odhadu zpoždění

2.4 Vizualizace dat

2.4.1 Funkční požadavky

•

- Aplikace vykreslí interaktivní mapu Prahy a širšího okolí, kterou bude možné posouvat či zoomovat. V této mapě budou zobrazeny vozidla na aktuálních pozicích a budou se automaticky posouvat po mapě, tak jak se pohybují ve skutečnosti.
- Po kliknutí na vozidlo se zobrazí jeho celá trasa včetně zastávek a jeho dopočítaného zpoždění.
- Po kliknutí na zastávku se zobrazí seznam spojů, které budou projíždět vybranou zastávkou a jejich trasy se vykreslí do mapy.
- Celá aplikace bude postavena na principu server – client. Tedy serverová strana se postará o přístup k otevřeným datům o vozidlech a jejich uložení a také obsluhu požadavků klienta. Klientská část bude webová stránka poskytující služby popsané výše. Měla by být schopná zobrazit řádově tisíce vozidel.

Nefunkční požadavky

- Serverová část bude napsaná v jazyce Python 3.
- Webová část bude napsaná pomocí jazyků pro webové technologie, převážně v JavaScriptu.
- Pro vykreslení mapy bude využita služba Mapbox.
- Ukládání dat na serverové straně bude řešeno MySQL databází.
- Pro algoritmus odhadu zpoždění na základě historických dat budou využity různé knihovny pro jazyk Python 3. Zejména pak scikit-learn a alphashape.

Proces běhu aplikace

Jak je již zmíněno aplikace bude využívat historická data, tedy bude nutné nechat aplikaci tato data nějakou dobu sbírat. Pro efektivní odhad by bylo vhodné mít uložené historické polohy vozidel alespoň z uplynulých několika týdnů.

Avšak již v průběhu sběru dat může aplikace poskytovat základní službu a to vizualizování vozidel v mapě.

2.4.2 Poskytovatelé mapových podkladů

K takovému účelu nejlépe poslouží vykreslení aktuálních poloh vozidel do mapy, kde se po vyžádání uživatelem tyto data zobrazí.

Za účelem vytvoření dostatečně přívětivé uživatelské aplikace je nezbytné využít některého z poskytovatelů mapových podkladů a zanést do něj získané informace.

Jedním z těchto poskytovatelů je společnost Google, která má propracované mapové podklady a prostřednictvím služby Google Maps poskytuje pro tuto práci požadovanou službu. Další platformou je Mapbox, který poskytuje velmi podobné služby jako Google Maps. Nicméně narozdíl od Googlu využívá jako mapový podklad OSM otevřená geografické data. Protože smyslem práce je v co největší míře využít otevřená data je žádoucí využít právě Mapbox.

TODO dokumentace mapbox, zeptat se jestli je to vubec nutne rozebirat

3. Implementace

4. Vizualizace dat

Pro vizualizaci dat do webového rozhraní na mapový podklad je využito prostředí Mapbox. K tomu je dále potřeba vytvořit serverovou aplikaci pro komunikaci s webovou stránkou.

4.1 Klientská část

Tato webová stránka je napsána ve standardním HTML s tím, že pro stylování objektů je použit jazyk CSS. Hlavní vlastnosti stránky, jako je zobrazení entit do mapy je použitý jazyk JS, zejména pak jeho možností pro zacházení s DOM elementy. Pro připojení a načítání dat ze serveru se používá technologie AJAXových dotazů.

Koncepce klientské aplikace je taková, že žádná data nezpracovává ani nepře-počítává a zobrazuje jen data taková, která obdržela od serverové strany typicky ve formátu GEOJSON.

Mapbox API

Prostředí Mapbox je široce využívaný multiplatformový nástroj pro zobrazení mapového podkladu a umožňuje do něj zanést širokou škálu různých geometrických útvarů. Takže mapové prostředí intuitivně interaguje s uživatelem a vývojáři mohou využít jednoduchého API pro zobrazení žádoucích dat do mapy.

Webová aplikace této práce využívá naprosto základní funkcionality, které mapbox umožňuje. Jejich popis včetně načtení do webové stránky za předpokladu, že jsou splněny základní parametry webové stránky je následující.

Rozhraní se do webové stránky importuje pomocí:

```
<script src='https://api.tiles.mapbox.com/
    mapbox-gl-js/v1.4.0/mapbox-gl.js'></script>
<link href='https://api.tiles.mapbox.com/
    mapbox-gl-js/v1.4.0/mapbox-gl.css' rel='stylesheet' />
```

Dále je potřeba vytvořit element s identifikátorem webové stránky, kde bude mapa zobrazena.

Po naimportování je v JavaScriptu k dispozici knihovna jménem `mapboxgl` pomocí, které se ovládá celé mapové prostředí. Tedy je tedy možné vytvořit samotnou mapu.

```
var map = new mapboxgl.Map({
    container: 'map', // identifikátor HTML elementu
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [14.42, 50.08], // střed mapy při
        inicializaci [lng, lat]
    zoom: 10 // zoom při inicializaci
});
```

Nyní stačí jen vytvořit HTML element za pomocí JS a po té může být přidám do mapy funkcí:

```
new mapboxgl.Marker(element)
    .setLngLat([Lng, Lat]) // zeměpisná výška a šířka
        umístění elementu
    .addTo(map);
```

Pro vykreslení složitějších objektů, jako je třeba lomená čára se využívá funkce `addLayer`.

```
map.addLayer({
    "id": id, // identifikátor vrstvy
    "type": "line", // geometrický útvar k zobrazení
    "source": {
        "type": "geojson", // formát zdrojových dat
        "data": data // zdroj dat
    },
    "paint": {
        "line-color": "#BF93E4", // barva
        "line-width": 5 // šířka
    }
});
```

K manipulaci s objekty typu `Layer` se používá

```
map.getLayer(id);
map.removeLayer(id);
```

Funkce a design aplikace

TODO vyspat co vsechno to umí, jak to udelat aby se to nekrylo s funkcnimi pozdavky??? + popsat design s obrazky

4.2 Serverová část

Příchozí požadavky od klineta jsou řešeny skriptem na serverové straně. Který je napojený na databázi a z ní extrahuje potřebná data.

Data jsou posílána v textové podobě ve formátu `GEOJSON`, který skrip konstruuje z dat získaných z databáze.

Server reaguje na 4 typy požadavků:

- `get_vehicle_positions` vrátí aktuální polohy všech vozidel,
- `get_tail` vrátí lomenou čáru popisující pohyb vozidla v uplynulých n minutách podle id traktu,
- `get_shape` vrátí lomenou čáru popisující trasu spoje podle id spoje,
- `get_stops` vrátí seznam zastávek pro spoj podle jeho id.

Server je naprogramován pomocí Pythonové knihovny `simple_server`, která slouží pouze k debugování, jak se píše v její dokumentaci. (TODO odkaz na dokumentaci). Protože se nepočítá s reálným nasezením této aplikace, není potřeba programovat robustní server. Pro demonstrační účely je však toto řešení dostatečné.

5. Algoritmus odhadu zpoždění

Tato kapitola popisuje hledání optimálního modelu pro popis pohybu vozidel na trase.

Z toho jak je problém formulován vyplývá, že se má odhadovat zpoždění mezi dvěma referenčními body a jediné takové jsou zastávky na trase daného spoje. Proto cíl algoritmu může být formulován jako vytvořit popis průběhu trasy mezi každou dvojicí zastávek, které alespoň jeden spoj obsluhuje a jsou bezprostředně sousedící ve sledu zastávek ve směru jízdy tohoto spoje. Nechť se dvojce bodů a spoj je obsluhující splňující předcházející předpoklad označuje jako A, B a S.

5.1 Základní předpoklady

Hned na začátek je potřeba ustanovit základní předpoklady ze kterých bude vycházet sestrojený algoritmus.

Zastávky je potřeba rozlišovat na jednotlivá nástupiště. Toto výrazně nezvýší počet dvojic zastávek A a B. – Naprostá většina zastávek má pouze dvě nástupiště. Pro každý směr jedno. Pokud má více nástupišť, pak tak bývá v případech, kdy ze zastávky odjízdí spoje do více směrů a tudíž pro každé nástupiště je jiná následující zastávka.

Všechny spoje S bez ohledu na linku nebo dopravce jedou ze zastávky A do zastávky B po stejně trase a tedy vzdálenost je konstatní. – Předpokládá se, že žáný dopravce nevyužívá jinou komunikaci a pro všechny platí pravidla silničního provozu stejně.

Čas jízdy ze zastávky A do B závisí pouze na denní době. – Vysvětleno výše. Navíc platí žádný z dopravců nedisponuje právem přednosti v jízdě před jiným dopravcem nebo výrazně výkonějším vozidlem. Dojezdové časy mohou být ovlivněny jen charakterem řidiče, avšak toto není zjistitelné z poskytnutých dat a zároveň se předpokládá, že charatery řidičů jsou rovnoměrně rozloženy mezi všechny dopravce a linky. Podle jízdních řádů některé linky jedou ve stejnou denní dobu rychleji než jiné, avšak skutečné doba jízdy je stejná. To že některý spoj zastávku projíždí a tím je rychlejší než jiný spoj není porušení tohoto předpokladu protože se jedná o dvě různé dvojice zastávek.

Během jízdy mohou nastat mimořádnosti, které porušují výše uvedené předpoklady, nicméně detekce mimořádností a jejich řešení je nad rámec této práce a jejich počet je zanedbatelný, proto na statistické modely nebudu mít vliv.

5.1.1 Analýza dat

Z dat popsaných v kapitole (TODO link kap 2???) je potřeba získat všechny dvojce zastávek A, B a všechny označené polohy vozidel mezi nimi. Toto je možné z databáze popsané v kapitole (TOTO link pak 3) zjistit pomocí jízdních řádů a dále pomocí atributu `dist_shape_traveled` uvedeného u každé zastávky pro každý spoj získat i jednotlivé polohy vozidel.

Toto je možné realizovat pomocí následujícího SQL dotazu.

TODO upravit a vylepsit tento dotaz, do jake miry mam uvadet vsechny pouzite sql dotazy???

```
SELECT schedule.id_trip,
       schedule.id_stop,
       schedule.lead_stop,
       departure_time,
       schedule.lead_stop_departure_time,
       (schedule.lead_stop_shape_dist_traveled - schedule.shape_dist_traveled)
           AS diff_shape_trav,
       trip_coordinates.inserted,
       (trip_coordinates.shape_dist_traveled - schedule.shape_dist_traveled)
           AS shifted_shape_trav,
       trip_coordinates.delay
FROM (
    SELECT id_trip, id_stop, shape_dist_traveled, departure_time,
           LEAD(id_stop, 1) OVER (
               PARTITION BY id_trip ORDER BY shape_dist_traveled) lead_stop
    LEAD(shape_dist_traveled, 1) OVER (
               PARTITION BY id_trip ORDER BY shape_dist_traveled) lead_stop
    LEAD(departure_time, 1) OVER (
               PARTITION BY id_trip ORDER BY shape_dist_traveled) lead_stop
    FROM rides) AS schedule
JOIN trip_coordinates
ON trip_coordinates.id_trip = schedule.id_trip AND
   schedule.lead_stop_shape_dist_traveled - schedule.shape_dist_traveled > 1500
   trip_coordinates.shape_dist_traveled BETWEEN schedule.shape_dist_traveled AND
   schedule.lead_stop_shape_dist_traveled
ORDER BY id_stop, lead_stop, shifted_shape_trav
```

Nyní je pro představení si situace uvedeno několik vizualizací těchto dat. Zejména pak s důrazem na rozlišnosti mezi průběhy tras mezi různými dvojicemi zastávek.

Všechny diagramy níže a posléze algoritmy pracují s těmito daty jako body třírozměrného prostoru. Každý tento bod reprezentuje jedno hlášení o poloze vozidla. První dimenze bodu je denní doba (v grafech znázorněno jako počet sekund od půlnoci), druhá dimenze je vzdálenost od výchozí zastávky (znázorněno jako počet metrů) a třetí dimenze reprezentuje čas od vyjetí z výchozí stanice (počet sekund).

TODO obrazky krátka trasa s moc nekonzistentními daty, dlouha trasa hladka, dlouha trasa s anomaliemi, trasy s nedostatkem dat, trasy kde jsou pekne videt krizovatky (cca 5 prikaldů)

5.2 Návrh modelu

Z výše uvedených vizualizací dat vyplývá, že hledání univerzálního modelu popisující všechny možné průběhy tras je nereálné.

Lepší je různé průběhy tras rozdělit podle jejich vlastností do kategorií a pro každou použít jiný model. Použité dílčí modely jsou:

5.2.1 Lineární model

Ačkoli je snaha tento model nahradit lepším, v některých situacích může jeho použití i na dálé dávat smysl. Zejména pak v případech kdy není k dispozici dostatek dat a nebo je vzdálenost dvou zastávek natolik malá, že nemá smysl ani jakýkoliv odhad zpoždění dělat. (TODO do problemu: Lepší by bylo volit trasy s nejmenší dobou jízdy, ale čas jízdy je proměnlivý a težko se získá skutečná doba jízdy z dat. Navíc v praxi jsou vzdálenost a doba jízdy dostatečně závislé)

(TODO obrázek lineárního modelu)

5.2.2 Polynomiální model

Polynomiální model se hodí pro situace kdy je průběh trasy nějak ovlivněn vždy ve stejném úseku a má vliv na každý projíždějící spoj. Nebo se v průběhu dne pozvolna mění v závislosti na dopravním vytížení projížděných úseků.

K tomuto dochází například v případech kdy spoj zastavuje ve městě a v následujících několika málo kilometrech jede pomaleji, poté zrychlý a dále opět vjede do města. Takový model se hodí spíše na delší trasy s plynulou jízdou.

Pro spočítání polynomiálního modelu se využívá knihovna sklearn konkrétně algoritmus zvaný Rigde, který sám o sobě hledá lineární závislosti, nicméně vstupní hodnoty jsou mezi sebou náležitě pronásobeny tak, aby simulovali polynomiální funkci. Toho se dosáhne pomocí funkce PolynomialFeatures. Optimální stupeň polynomu se zjistí spočítáním modelu pro každý stupeň v rozumných mezích a nakonec se zvolí ten s nejmenší chybou. (TODO jak moc detailně se ma toto popisovat?)

(TODO obrázek poly modelu)

Jako odhad zpoždění se pak vrací rozdíl skutečného počtu sekund na trase a predikce modelu. To celé se pak ještě přičítá k rozdílu predikce v modelu v čase a vzdálenosti příjezdu podle jízdního řádu a pravidelného příjezdu.

5.2.3 Model pomocí konkávního obalu

Posledním a nejkomplikovanějším modelem, zasluhující nejvíce pozornosti, je popsání trasy za pomocí konkávního obalu bodů.

Tato metoda vznikla jako řešení situace, kdy na trase exituje bod, který určité procento projíždějících spojů zdrží o netriviální dobu. Něco takového nastane pokud spoje projíždí světelnou křižovatkou nebo místem přek kterým se tvoří kolona vozidel. Zde dochází ke skové změně průběhu bodové funkce a spojité modely by s okolím tohoto kritického místa měly problém. Pro jeden takový bod by možná šlo navrhnut jednoduší řešení, nicméně je potřeba algoritmus, který umí pracovat s více kritickými body na trase. Například je nutné poradit si se situací zobrazené na diagramu (TODO link na obrázek), kde jsou na trase takové

kritické body dva. V této modelové situaci je pro jednoduchost uvažováno, že se většina projíždějících spojů zdrží v prvním kritickém bodě, nebo v druhém, nebo se lehce zdrží v obou. S takovým zdržením je počítáno v jízdním řádu a tedy vozidla, která projedou první kritický bod bez zdržení jedou na čas stejně tak, jako vozidla v něm zdržená. O snížení nebo zvýšení případného zpoždění spoje je možno rozhodnout až po projetí druhého bodu. Jinými slovy na ose uplynulého času od vyjetí ze zastávky vzniká jakýsi podprostor v němž se zpoždění nemění. Pro jeho popis se využívá právě konkávní obal všech spojů, které přijely včas.

Problémy

Ačkoli je myšlenka jednoduchá, při implementaci vyvstává několik technických problémů.

Nejprve k samotnému konkávnímu obalu. Je potřeba říct, že na množině bodů není definován jednoznačně (TODO obrazek nejednoznačnosti konkavnoho obalu). A jedná se o početně složitý algoritmus. V tomto díle je zapotřebí spočítat obal ve třídimenzionálním prostoru, což je velmi komplikovaný úkol, a proto je potřeba přijít s zjednodušením úlohy. Tedy počítat obal pouze pro dvoudimenzionální prostor. Toho se nedá dosáhnou jinak než diskretizací úlohy a počítání obalu pro každou hodinu zvlášt̄, tedy ze všech bodů, které byly zaznamenány v průběhu jedné hodiny. Tím může dojít k vetší granularitě obalu, než by bylo vhodné, nicméně předpokládá se, že hodina je dostatečně dlouhý časový interval na to, aby zde byly zachyceny všechny druhy průběhu jízdy (započítávají se všechny spoje jedoucí ve stejnou hodinu nejméně týden zpět) a zároveň je to dostatečně krátký interval na nezkreslování denních výkyvů v čase jízdy. Navíc všechny body tvořící obal pro danou hodinu jsou poté přidány do celkového obalu i s přesným časem zaznamenáním a pro počítání konečných výsledků je použit třídimenzionální obal.

Dále se jako netriviální ukazuje detekce spojů, které přijely včas a tedy všechny jejich body mají být předány k výpočtu obalu. Nabízí se použít data o všech spojích, které přijely do cílové zastávky s co nejmenším zpožděním, ale je nutné mít na paměti, že příjezdy podle jízdního řádu nemusí vůbec odpovídat realitě. Proto se zdá být nejlepší použít data od spojů, které přijely ve stejnou dobu jako je průměr všech příjezdů do cílové zastávky. Toho se docílí tak, že se poslední body podle vzdálenosti všech spojů použijí pro odhad času příjezdu. Zobrazeno na diagramu (TODO link na obrazek odhad příjezdu). Dále se pro každou hodinu seřadí všechny spoje podle vzdálenosti odhadnutého příjezdu od skutečného a použije se určité procento nejbližších spojů k tomuto odhadu.

Z předchozího popisu řešení vyplývá ovšem, že pro výpočet obalu jsou použity spoje, které ani zdaleka nemusely přijet včas jak je požadováno, ale předpokládá se, že se nepříliš vzdalují od průměrného času příjezdu. To že střední zpoždění pro celý obal není nulové se vyřeší spočtením odchylky průměrného příjezdu od příjezdu podle jízdního řádu a následně přičtení této konstanty k odhadnutnému zpoždění. Každopádně to, že rozptyl příjezdů spojů zahrnutých ve výpočtu obalu může být netriviální, vyžaduje nahlížet na tento obal jako na lineární prostor pohybu zpoždění. Tedy že odhad zpoždění pro bod nacházející se v obalu je lineárně

závislý na vzdálenosti od hranice obalu, avšak protože je známo časové rozpětí příjezdu spojů použitých pro výpočet obalu, je možné tuto vzdálenost snadno přenést na skutečné pozdění.

Popis algoritmu

Po popsání a vyřešení všech komplikací je možné nastínit průběh algoritmu.

Nejprve konstrukce konkávního obalu:

```
poslední_čas = vyber všechny poslední body podle vzdálenosti od každého spoje  
odhad_příjezdu = odhadni čas příjezdu v průběhu celého dne podle bodů v poslní_čas  
spoje_včas = prázdné pole
```

```
pro každou hodinu h:
```

```
    spoje_včas += vyber x \% spojů, které přijely nejblíže odhadu v hodině h
```

```
konkávní_obal = prázdné pole
```

```
pro každou hodinu h:
```

```
    body = vyber všechny body zaznamenané v hodině h a náležící kterémukoli spoji v  
    konkávní obal += spočítej konkávní obal z body
```

```
Vrací: konkávní_obal
```

Dále odhad zpoždění z konkávního obalu:

```
Vstup: bod v prostoru vzdálenosti, průběhu dne a času na trase
```

```
pokud je bod v konkávní_obal:
```

```
    velikost_okna = rozdíl horní hranice obalu od spodní v čase příjezdu do zastávky  
    spodek_okna = spodní hranice okna v čase příjezdu do zastávky  
    poměr = vzdálenost bodu od spodní hranice obalu / vzdálenost bodu od horní hranice  
    odhad_příjezdu = velikost_okna * poměr + spodek_okna
```

```
jinak:
```

```
    pokud je bod pod obalem:
```

```
        spodek_okna = spodní hranice obalu v čase příjezdu do zastávky  
        odhad_příjezdu = spodek_okna - vzdálenost bodu od obalu
```

```
    jinak:
```

```
        vrch_okna = horní hranice obalu v čase příjezdu do zastávky  
        odhad_příjezdu = vrch_okna + vzdálenost bodu od obalu
```

```
Vrací: odhad_příjezdu - pravidelný příjezd
```

6. Porovnání se stávajícím řešením

TODO tabulka pro kazdy model s porovnanim odhadu zpodeni nyni a podle meho modelu

7. Navrhy na zlepšení

TODO

8. Statistiky

TODO neni to primo v zadani prace, ale ruzne statistiky nad zpozdenima by mohly byt zajimave, je to vhodne???

Závěr

Seznam použité literatury

Seznam obrázků

1.1	Graf počtu úseků mezi následujícími zastávkami a vzdálenotí mezi nimi.	6
1.2	Modrá plocha značí vymodelovaný profil trasy. Oražové body jsou jednotlivé vzorky polohy vozidel. Data pro graf jsou ze dnů 20.–21. 2. 2020	7
1.3	Trasa mezi zastávkama K Letišti a Zličín. Zdroj: mapy.cz	7
1.4	Mapa z golemio.cz.	15
1.5	Mapa z www.tram-bus.cz.	15
1.6	Mapa z mapa.idsjmk.cz.	16
2.1	UML diagram návrhu aplikace	18
2.2	EER diagram databáze.	20

Seznam tabulek

Seznam použitých zkratek

Slovník

- AJAX** Asynchronous JavaScript and XML. 26
- API** rozhraní pro programování aplikací. 9, 26
- CSS** Cascading Style Sheets. 26
- DOM** Document Object Model. 26
- DPP** Dopravní podnik hlavního města Prahy, a.s. 8, 14
- GEOJSON** standardní formát navržený pro reprezentaci jednoduchých prostorových geografických dat, specifikace: <https://tools.ietf.org/html/rfc7946>. 9, 22, 26, 27
- GPS** Global Position System. 8, 10, 12
- HTML** Hypertext Markup Language. 26, 27
- HTTP** HyperText Transfer Protocol. 9
- IDSJMK** Integrovaný dopravní systém Jihomoravského kraje. 16
- INT** Celé číslo. 21
- JS** JavaScript. 26, 27
- JSON** JavaScript Object Notation, specifikace: <https://tools.ietf.org/html/rfc8259>. 9, 19
- Kapsch** Kapsch, rodinná firma. 8
- MHD** městská hromadná doprava. 4, 11
- OSM** OpenStreetMap. 24
- ROPID** Regionální organizátor pražské integrované dopravy, p. o.. 3, 9
- SQL** Structured Query Language. 19, 30
- URL** Unique Resource Link. 9
- UTC** Koordinovaný světový čas. 10
- VHD** Veřejná hromadná doprava. 14

A. Přílohy

A.1 První příloha