Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Vyhledávač dopravních spojů veřejné dopravy v Olomouci

2024

Bc. Stanislav Cingel

Vedoucí práce:
RNDr. Martin Trnečka, Ph.D.

Studijní program: Aplikovaná informatika,
Specializace: Vývoj software

Bibliografické údaje

Autor:	Bc. Stanislav Cingel
Název práce:	Vyhledávač dopravních spojů veřejné dopravy v Olomouci
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Aplikovaná informatika, Specializace: Vývoj software
Vedoucí práce:	RNDr. Martin Trnečka, Ph.D.
Počet stran:	66
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	slovenský

Bibliographic info

Author:	Bc. Stanislav Cingel
Title:	Journey planner for public transport in Olomouc
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Applied Computer Science, Specialization: Software Development
Supervisor:	RNDr. Martin Trnečka, Ph.D.
Page count:	66
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Slovak

Anotácia

Práca sa zaoberá vytvorením vyhľadávača spojov verejnej dopravy pre mesto Olomouc. Pozostáva zo získania a spracovania dát nielen pre Olomouc, ale aj zvyšok Českej a Slovenskej republiky z verejných aj neverejných zdrojov. Základom nového vyhľadávača je vyhľadávač Najdispoj, ktorého stav je v práci zhodnotený a sú uvedené jeho hlavné nedostatky. Na základe tejto analýzy sú v klientskej aj serverovej časti aplikácie implementované potrebné úpravy. Práca skúma aj spôsob nasadenia aplikácie do prevádzky a hardvérové nároky jej rôznych konfigurácií. Výsledkom je vylepšený vyhľadávač spojov, ktorý je verejne dostupný na internete. V závere práce je naznačených niekoľko možných smerov ďalšieho rozvoja projektu.

Synopsis

The thesis deals with the creation of public transport journey planner for the city of Olomouc. It consists of obtaining and processing data not only for Olomouc, but also for the rest of the Czech Republic and Slovakia from both public and non-public sources. The basis of the new journey planner is the Najdispoj journey planner, the status of which is evaluated in the work and its main shortcomings are listed. Based on this analysis, the necessary modifications are implemented in both the client and server parts of the application. The work also examines the way the journey planner is deployed and the hardware requirements of its different configurations. The result is an improved journey planner that is publicly available on the internet. In the conclusion, several possible directions for the further development of the project are indicated.

Kľúčové slová: vyhľadávač dopravných spojov; open data; webová aplikácia; OpenStreetMap; OpenTripPlanner; GTFS

Keywords: journey planner; open data; web application; OpenStreetMap; OpenTripPlanner; GTFS

Za príležitosť pracovať na téme podľa vlastného výberu ďakujem vedúcemu tejto diplomovej práce RNDr. Martinovi Trnečkovi, Ph.D. Za ochotu a poskytnutie dát ďakujem spoločnosti *Dopravní podnik města Olomouc a.s.* Nesmierne si vážim nekonečnú podporu a motiváciu od mojej rodiny a priateľov počas písania tejto práce a celého štúdia. V neposlednom rade chcem poďakovať všetkým ľudom, ktorí nezištne prispievajú k rozvoju open source projektov.

Odovzdaním tohto textu jeho autor|ka čestne vyhlasuje, že celú prácu vrátane príloh vypracoval|a samostatne a za použitia iba zdrojov spomínaných v texte práce a uvedených v zozname literatúry.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam zdrojových kódov

1 Úvod a motivácia

V tejto práci budem čiastočne nadväzovať na svoju bakalársku prácu.[1] Tá sa zaoberala vytvorením vyhľadávača spojov verejnej dopravy pre mesto Bratislava. Vyhľadávač obsahoval jediný zdroj dopravných dát, a to dáta *Dopravného podniku Bratislava a.s.*. Postupne boli však zverejňované ďalšie zdroje dát a s tým prišla potreba upraviť aplikáciu tak, aby bola schopná tieto zdroje spracovať a využívať.

Zadaním práce je vytvoriť vyhľadávač spojov verejnej dopravy pre mesto Olomouc. Preto bolo primárnou snahou získať dáta z *Dopravného podniku města Olomouce*, avšak bola preskúmaná aj dostupnosť ďalších zdrojov dát nielen z okolia mesta Olomouc, ale aj zo zvyšku Českej republiky a Slovenska.

Vyhľadávač spojov je nástroj umožňujúci cestujúcim nájsť najvhodnejšie spôsoby dopravy medzi dvoma bodmi s použitím spojov verejnej dopravy. Okrem štandardných vstupov, ako počiatočná a konečná zastávka, či čas odjazdu/príjazdu, môžu zohľadňovať špecifické požiadavky na prepravu daného cestujúceho – preferencia dlhšieho pešieho presunu namiesto prestupu, obmedzenie len na linky s nízkopodlažnými vozidlami a tak podobne. Funguje v niekoľkých fázach:

1. **Získanie a spracovanie geografických a dopravných dát.** Jedná sa o geografické dáta v celej oblasti pokrytej sieťou liniek verejnej dopravy + okolie zastávok v dostatočnej pešej vzdialenosti. Vyhľadávače môžu fungovať aj bez geografických dát, čo sa ale odzrkadlí na kvalite výsledkov vyhľadávania (napríklad absencia presných peších presunov, výsledky bez prehľadného zobrazenia na mape).

2. **Spracovanie dát do vhodnej dátovej štruktúry.** V závislosti na použitom vyhľadávacom algoritme sa môže jednať o graf, alebo inú dátovú štruktúru. Tento a predchádzajúci krok nie je závislý na používateľskom vstupe, a teda ich stačí vykonať iba pri zmene vstupných dát.

3. **Vyhľadanie najlepšej cesty na základe zadania.** Po tom, ako používateľ zadá štart, cieľ a ostatné parametre vyhľadávania, sa spustí vyhľadávací algoritmus, ktorý vráti (ak existuje) najlepšiu cestu spĺňajúcu požiadavky.

4. **Zobrazenie výsledkov.** Výsledky vyhľadávania sa zobrazia používateľovi v prehľadnej forme. Okrem základných informácií o spojoch (čas odjazdu/príjazdu, dĺžka cesty, počet prestupov) sa môže zobraziť aj mapa s trasou cesty, či napríklad informácie o zastávkach, ktorými cesta prechádza.

1.1 Štandardné formáty dopravných dát

1.1.1 General Transit Feed Specification (GTFS)

GTFS je štandard pre výmenu dát o verejnej doprave od spoločnosti *Google*, pôvodne pre potreby aplikácie *Google Maps*. Obsahuje informácie o zastávkach, lin-

[1] https://theses.cz/id/ptxld7/

kách, spojoch, cestovných poriadkoch, tarifách a ďalších údajoch. Je primárne určený na použitie vo vyhľadávačoch spojov [1].

Je tvorený niekoľkými súbormi vo formáte CSV zabalenými do ZIP archívu. Jednotlivé súbory modelujú konkrétne aspekty cestovných poriadkov – dopravcov, zastávky, linky, trasy a tak ďalej.

Formát CSV bol zvolený preto, že je ho možné jednoducho upravovať akýmkoľvek textovým editorom. Tým sa zníži bariéra účasti natoľko, že GTFS dáta mohli vytvárať aj dopravné spoločnosti s menšími prostriedkami a bez špecializovaného softvéru [2].

1.1.2 Network Timetable Exchange (NeTEx)

NeTEx je CEN štandard dopravných dát, využívajúci XML formát na popis dopravnej siete [3]. Dáta v tomto formáte môžu slúžiť aj na iné účely než vyhľadávanie spojov. NeTEx môže byť použitý aj počas procesu tvorby cestovných poriadkov, je ním možné popísať aj zložité dopravné systémy a tarify. Dáta vo formáte NeTEx je možné stratovo prekonvertovať do formátu GTFS [4].

1.1.3 Jednotný datový formát (JDF)

JDF je formát využívaný v Českej republike a na Slovensku. Na základe pokynu Ministerstva dopravy Českej republiky [5] sú v tomto formáte poskytované dáta systémom CIS JŘ (viď sekcia 3.1.3).

Jedná sa o sadu textových súborov zabalených do ZIP archívu. Tieto textové súbory využívajú formát CSV s bodkočiarkou na konci riadku [6]. Popis formátu (vo verzii 1.10) je dostupný na adrese.[2]

K dispozícii je niekoľko nástrojov na konverziu dát z formátu JDF do formátu GTFS:

- **JrUtil** – nástroj na konverziu dát vytvorený Davidom Koňaříkom.[3] Jeho použitie v súvislosti so zdrojom dát CIS JŘ je popísané v sekcii 3.1.3.
- **jdf2gtfs** – dlhšie neaktualizovaný nástroj od autora Jana Masopusta.[4] Podobne ako *JrUtil* dokáže ako zdroj dát použiť CIS JŘ.

1.1.4 GTFS Realtime

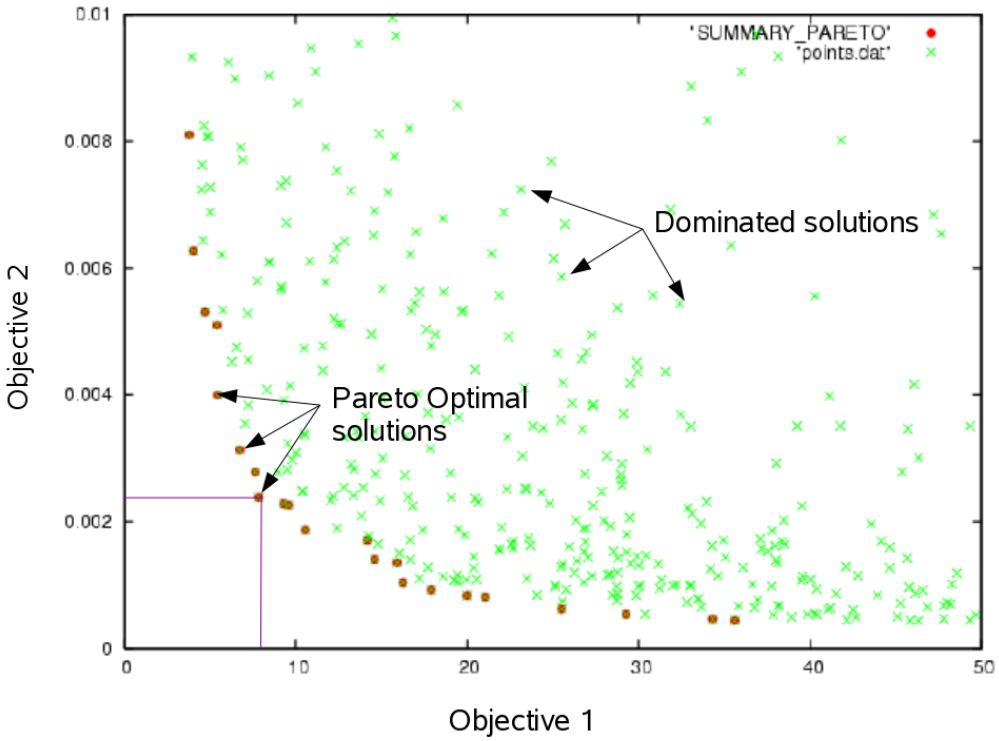
Jedná sa o rozšírenie formátu GTFS, umožňujúce dopravným spoločnostiam poskytovať v reálnom čase dáta súvisiace s dopravnou sieťou – polohy a meškanie vozidiel, zrušenie spoja, nečakané udalosti ovplyvňujúce zastávku, trasu, či celú sieť [7]. Je založený na tzv. *Protocol Bufferoch* – mechanizme serializácie štruktúrovaných dát [8].

[2] https://chaps.cz/files/cis/jdf-1.10.pdf
[3] https://github.com/dvdkon/jrutil
[4] https://github.com/masopust/jdf2gtfs

1.2 Vyhľadávacie algoritmy

V rámci práce je na samotný proces vyhľadávania spoja prostredníctvom algoritmu využívaný nástroj *OpenTripPlanner*. Pôvodný pokus o implementáciu vlastného vyhľadávacieho algoritmu je stručne popísaný v bakalárskej práci, v skratke však možno povedať, že sa nejednalo ani zďaleka o riešenie konkurujúce existujúcim open source riešeniam, ktoré sú vytvárané a udržiavané expertmi z oblasti verejnej dopravy. Napriek tomu je však pre prehľad vhodné uviesť aspoň v zjednodušenej forme niekoľko základných pojmov a vyhľadávacích algoritmov. Pri popise algoritmov budem parafrázovať publikáciu *Round-Based Public Transit Routing*[5] od autorov Daniel Delling, Thomas Pajor a Renato F. Werneck.

1.2.1 Paretovo optimum



Obr. 1: Znázornenie Pareto množiny. Zdroj: https://web.archive.org/web/20200226003108/http://www.cenaero.be/Page.asp?docid=27103&

Paretovo optimum je pojem pochádzajúci z ekonómie, ktorý našiel využitie aj v oblasti informatiky v kontexte optimalizácie. Jedná sa o stav, v ktorom nie je možné

[5] https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf

zvýšiť uspokojenie jednotlivca bez toho, aby sa znížilo uspokojenie druhého [9]. V prípade vyhľadávania spojov sa jedná zvyčajne o nájdenie Paretovho optima s minimálnym časom príjazdu a s minimálnym počtom prestupov. Obr. 1 znázorňuje množinu Paretovych optím.

Vyhľadávanie spojov je teda prirovnateľné k hľadaniu tzv. *Pareto množiny* – množiny všetkých Pareto-optimálnych riešení.

1.2.2 Dijkstrov algoritmus

Ak sa rozhodneme na vyhľadávanie všetkých Pareto-optimálnych spojov pozerať ako na grafový problém, kde minimalizujeme čas príjazdu, môžeme ho riešiť pomocou variantov Dijkstrovho algoritmu. V najjednoduchšej forme tohto, kedysi bežného algoritmu na vyhľadávanie spojov, sú za vrcholy grafu považované odjazdy a príjazdy na konkrétnu zastávku v konkrétnom čase, zatiaľ čo hrany reprezentujú čakanie medzi dvoma udalosťami na konkrétnej zastávke alebo presun medzi dvoma zastávkami. Jedná sa o tzv. *time-expanded* model [10]. Tento model však spôsobuje, že zostrojený graf je príliš veľký na efektívne vyhľadávanie.

Time-dependent model komplexnosť grafu znižuje tým, že zoskupuje cesty medzi zastávkami do časovo závislých funkcií. Veľkosť takto zostrojeného grafu je lineárna voči počtu zastávok a liniek a rádovo menšia než v prípade *time-expanded* modelu [11]. Je ale potrebné použiť upravenú, zložitejšiu verziu Dijkstrovho algoritmu (Time-Dijkstra). Na to, aby sme mohli tento model použiť s dvoma alebo viacerými kritériami (napríklad počet prestupov, optimalizácia ceny, či iné arbitrárne kritérium), je treba použiť tzv. *multi-label-correcting (MLC)* algoritmus. V prípade, keď okrem času príjazdu chceme zohľadniť len jedno dodatočné kritérium, ktoré je diskrétne (napríklad počet prestupov), je možné použiť aj jednoduchší *Layered-Dijkstra (LD)* algoritmus [11]. Existuje viacero vylepšení a variantov uvedených algoritmov, ktoré je možné aplikovať – ich stručný popis sa nachádza v druhej kapitole vyššie uvedenej publikácie.

1.2.3 A*

Tento variant Dijkstrovho algoritmu sa využíva spolu s nižšie uvedenými algoritmami na nájdenie peších presunov od štartu k potenciálnym počiatočným zastávkam a od potenciálnych cieľových zastávok k cieľu.

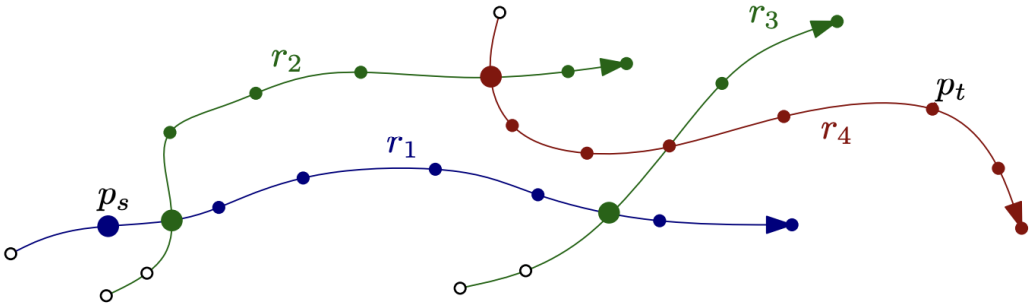
1.2.4 RAPTOR

Round-bAsed Public Transit Optimized Router je riešením problému hľadania spojov podľa dvoch kritérií, ktoré nevyužíva žiaden variant Dijkstrovho algoritmu, ani explicitný graf (napriek tomu, že objekt vygenerovaný nástrojom *OpenTripPlanner 2*, obsahujúci vstupné dáta pre tento algoritmus, je uložený v súbore *graph.obj*). Bol predstavený v publikácii *Round-Based Public Transit Routing* [11].

Vo svojej základnej verzii funguje na princípe cyklov. Algoritmus sa v k-tom cykle pokúša nájsť spoj z počiatočnej zastávky do cieľovej zastávky s $k-1$ prestupmi (zná-

zornené na obr. 2). Počas každého cyklu zváži každú linku maximálne raz (predmetom jednej z optimalizácií je vynechanie liniek, ktoré nie sú dosiahnuteľné v predchádzajúcom cykle).

Medzi varianty algoritmu patria *McRAPTOR*, ktorý zovšeobecňuje algoritmus pre dodatočné kritéria a *rRAPTOR*, ktorý ako vstupný parameter nevyužíva konkrétny čas, ale časový interval, pričom vráti kompletné Pareto množiny pre každý odjazd v danom intervale.



Obr. 2: Hľadanie cesty z p_s do p_t. Cesta r_1 je prehľadávaná v prvom cykle, r_2 a r_3 v druhom, r_4 v treťom. Zdroj: https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf

V porovnaní s algoritmom *RAPTOR* sú algoritmy *MLC* a *LD* z dôvodu využitia prioritnej fronty pomalšie aspoň o logaritmický faktor.

Veľkou výhodou tohto algoritmu je možnosť jednoduchej paralelizácie, čo umožňuje rýchlejšie vyhľadávanie spojov [11]. Zároveň nevyžaduje časovo náročné predspracovanie (napríklad budovanie orientovaného grafu), takže je použiteľný vo vysoko dynamických scenároch s výlukami a meškajúcimi spojmi.

1.3 Existujúce vyhľadávače spojov v Olomouci

V nasledujúcich podsekciách budú predstavené populárne vyhľadávače spojov, prostredníctvom ktorých je možné vyhľadávať spoje v olomouckej MHD.

1.3.1 IDOS

Hlavnou výhodou vyhľadávača IDOS[6] je aktuálnosť a rozsah použitých dát, keďže sa nachádza „pri prameni" (aplikácia je vytvorená spoločnosťou *CHAPS spol. s r.o.* – správcom systému CIS JŘ). Okrem vyhľadávania spojov ponúka aj možnosť nákupu cestovných lístkov (nedostupné v olomouckej MHD). Výhodou je aj možnosť vyhľadávania spojov v okolí (napríklad v rámci celého kraja či štátu). Štart a cieľ je možné zadávať aj pomocou mapy – nemôžu byť zadané presné súradnice, len názov zastávky, či adresa. Výsledky vyhľadávania je možné zobraziť aj na mape, pešie presuny sú zobrazené vzdušnou čiarou. Neobsahuje dynamické dáta olomouckej MHD.

1.3.2 Google Maps

Aplikácia *Google Maps* obsahuje dáta olomouckej MHD. Tie sú poskytované priamo spoločnosťou DPMO (jedná sa o rovnaký GTFS export, aký je umiestnený na ich webových stránkach). Podľa informácií od dopravného podniku, aktualizácia dát môže trvať niekoľko dní, dokonca až týždňov. Preto sa v prípade výluk na dáta nemožno spoliehať. Napriek tomu sa nedá poprieť, že používateľský zážitok je v prípade tejto aplikácie veľmi dobrý, čo súvisí nielen s prepracovaným zadávaním vyhľadávania, a prehľadným zobrazením výsledkov, ale aj s prepojenosťou s ostatnými službami spoločnosti *Google*.

1.3.3 Mapy.cz

Používatelský zážitok pri vyhľadávaní spojov je v aplikácii *Mapy.cz*[7] na podobnej úrovni ako v *Google Maps*. Nie je možné určiť, ako často sú dopravné dáta aktualizované, ale vzhľadom na český pôvod aplikácie a dlhodobo trvajúce súdne spory ohľadom poskytovania dát medzi spoločnosťami *CHAPS spol s r.o.* a *Seznam.cz, a.s.*[8] je možné predpokladať, že je kladený väčší dôraz na aktuálnosť a presnosť dát.

[6] https://idos.idnes.cz/

[7] https://mapy.cz/zakladni?planovani-trasy

[8] https://www.lupa.cz/aktuality/antimonopolni-urad-znovu-potvrdil-pokutu-pro-chaps-za-zadrzovani-dat-o-jizdnich-radech/

2 Pôvodný stav

Pod pôvodným stavom je myslený stav vyhľadávača v čase odovzdania bakalárskej práce (máj 2021), predmetom ktorej bolo jeho vytvorenie. V tejto kapitole budú popísané hlavné problémy a nedostatky, ktoré boli identifikované pri vývoji a po nasadení aplikácie.

2.1 Zdroje dát

Použité boli iba statické dáta *Dopravného podniku Bratislava, a.s.* v kombinácii s geografickými dátami *OpenStreetMap*. Na rozšíriteľnosť, či prípadnú možnosť nahradenia týchto zdrojov inými nebol počas vývoja kladený žiaden dôraz.

2.2 Klientska a serverová časť

Klientska časť bola napísaná vo frameworku *Vue.js 3*. Ku koncu jej vývoja sa ako hlavný problém javila absencia pokročilejšieho state managementu, čo sa prejavovalo spomaleným vývojom a ťažšou čitateľnosťou kódu. Celkovo ale je možné skonštatovať, že tieto problémy sú odstrániteľné aj bez kompletnej prestavby klientskej časti tak povediac „na zelenej lúke".

Serverová časť, napísaná v jazyku *Python*, využíva knižnicu *FastAPI*. Jedná sa v podstate o niekoľko *Python* súborov bez akejkoľvek štruktúry, otypovania, či testov. Jednoduchosť serverovej časti je daná už spomenutým nízkym počtom zdrojov dát a absenciou vízie rozširovať aplikáciu do budúcna (napríklad poskytovaním abstraktných rozhraní pre rôzne služby/zdroje dát). V tomto kontexte sa jednalo o dostatočné, avšak so zväčšujúcim sa rozsahom projektu neudržateľné riešenie.

Nasadenie aplikácie do produkcie prebiehalo jednoduchým spustením serverovej časti mimo akýkoľvek kontajner. Beh na pozadí aj po zatvorení terminálu zabezpečoval príkaz *screen*.

2.3 Identita vyhľadávača

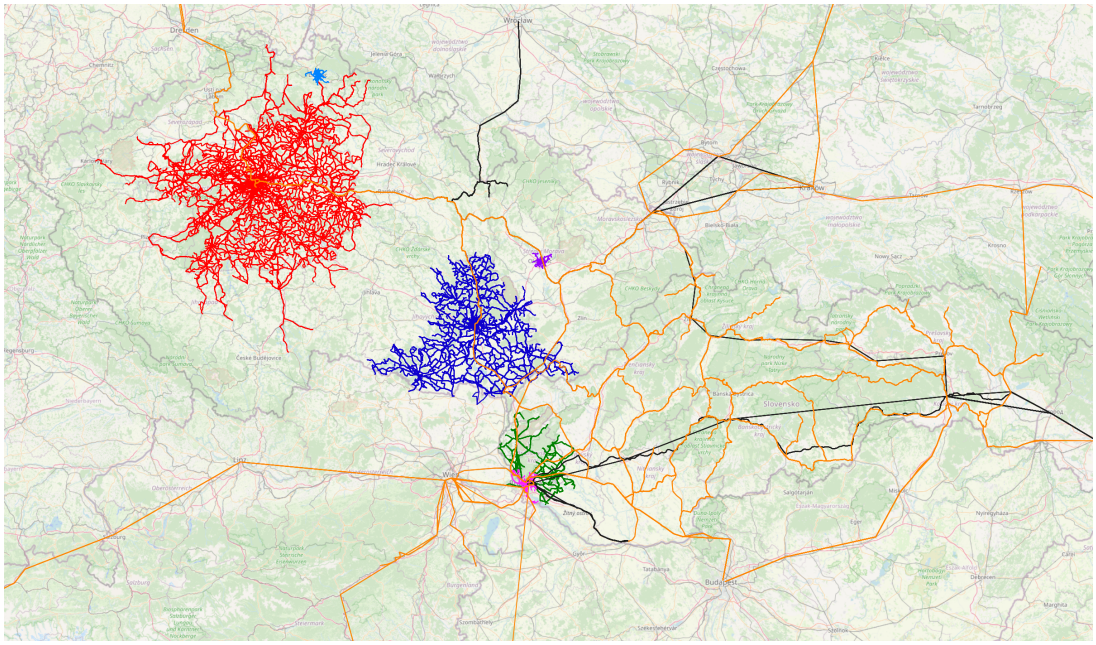
Jednotná identita produktu je kľúčovým faktorom pre jeho rozpoznateľnosť. Pôvodná verzia sa vyznačovala predovšetkým absenciou jednotného názvu. Medzi používané názvy patrili *Vyhľadávač NájdiSpoj.sk*, *Nájdispoj*, či *najdispoj.sk*. Keďže je pri novej verzii kladený väčší dôraz na vytváranie vlastných inštancií (na vlastných doménach), je potrebné odstrániť viazanosť na doménu *najdispoj.sk* a zvoliť univerzálnejší názov. Podobnou úvahou sa dá dospieť aj k odstráneniu diakritiky, ktorá viazala názov na konkrétny jazyk/abecedu. Zároveň už na internete existujú referencie na tento produkt a v určitých kruhoch je známy, preto úplná zmena názvu nepripadá do úvahy.

Po týchto úvahách bolo rozhodnuté pre nový jednotný názov *Najdispoj*.

3 Implementácia potrebných úprav

3.1 Zdroje dát

V nasledujúcich sekciách je popísaná väčšina úspešných aj menej úspešných pokusov o získanie dopravných dát. Popísaný je aj zdroj geografických dát – *OpenStreeteMap*. Získané dáta sú znázornené na obrázkoch 3 a 4.



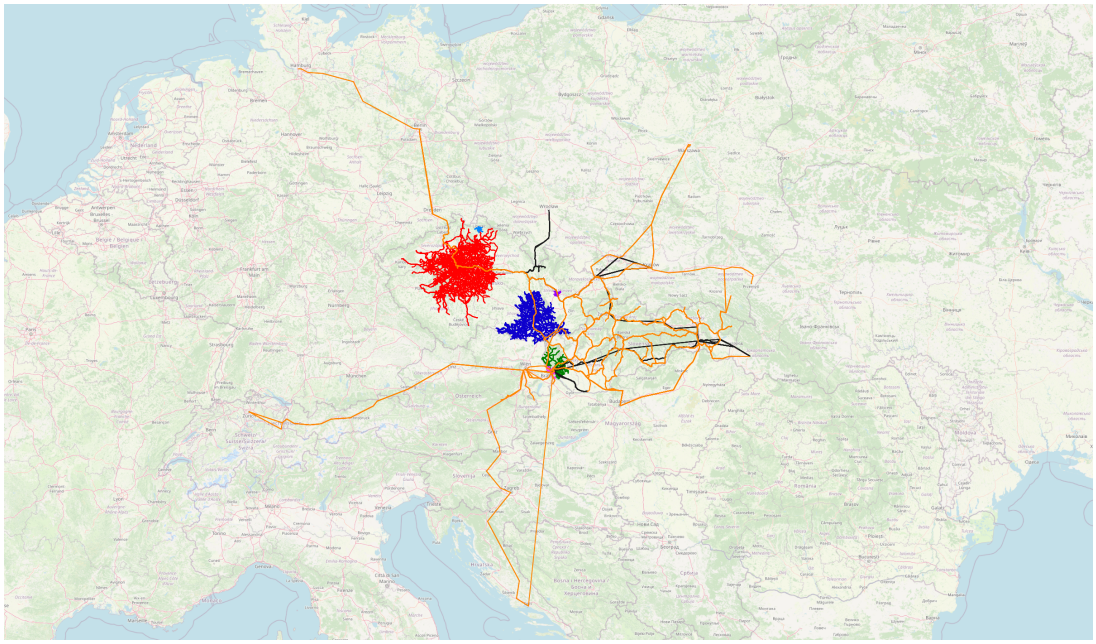
Obr. 3: Vizualizácia použitých zdrojov dát v ČR a SR. Mapové podklady: *OpenStreetMap*

3.1.1 Dopravní podnik města Olomouce, a.s. (DPMO)

Statické dáta boli pôvodne dostupné len vo formáte JDF, avšak niekoľko dní po požiadaní o sprístupnenie dát vo formáte GTFS na účely ich využitia v diplomovej práci boli zverejnené na webových stránkach DPMO.[9] Nové dáta sú zverejňované po každej zmene, vrátane výluk.

S dynamickými dátami je situácia zložitejšia. Interne v DPMO existujú dva druhy dynamických dát – dáta z chytrých zastávok pre zastávkové tabule a dáta pre dispečing. Ani jeden z nich však nie je dostupný verejne. V DPMO bola vyvinutá snaha o poskytnutie dát zo zastávkových tabúľ. Po vyriešení zmluvných záležitostí s poskytovateľom služieb rádiovej siete využívanej pre komunikáciu bolo potrebné umiestniť na centrálny bod siete (Svatý Kopeček) server, ktorý by dáta zhromažďoval a poskytoval k nim prístup. Tento krok však zatiaľ nebol uskutočnený.

[9] https://www.dpmo.cz/informace-pro-cestujici/jizdni-rady/jizdni-rady-gtfs/



Obr. 4: Vizualizácia použitých zdrojov dát s presahom mimo ČR a SR. Mapové podklady: *OpenStreetMap*

Napriek tomu mi ale bola poskytnutá vzorka dát (len centrum mesta) a ich popis. Keďže sa jedná o komunikáciu modemov vo vozidlách s modemami na zastávkach a s virtuálnymi bodmi medzi zastávkami a dáta neobsahujú presné súradnice, je možné určiť polohu vozidiel len podľa zastávok, s ktorými práve komunikujú. Dáta obsahujú čas meškania s presnosťou na minúty (využívané na zastávkových tabuliach). Mali by obsahovať dostatok informácií na spárovanie so statickými dátami, čo však nemožno s istotou potvrdiť bez väčšej vzorky dát. Snaha o získanie prístupu k týmto dátam bude pokračovať aj po dokončení diplomovej práce.

3.1.2 Integrovaný dopravní systém Olomouckého kraje (IDSOK)

Možnosť vyhľadávať spoje nielen v meste, ale aj v blízkom okolí (regionálny autobus + prestup na mestskú hromadnú dopravu) značne zvýši využiteľnosť vyhľadávača. Preto som sa pokúsil aj o získanie dát z celého IDSOK. So žiadosťou som najprv oslovil KIDSOK (*Koordinátor Integrovaného dopravního systému Olomouckého kraje, p. o.*). Po zamietnutí žiadosti, odôvodnenom extrémnou pracovnou vyťaženosťou, som sa obrátil na námestníka pre oblasť dopravy Olomouckého kraja Michala Záchu DiS., ktorému bola v tom čase organizácia KIDSOK priamo podriadená. Odpoveď však bola podobného charakteru.

3.1.3 Celostátní informační systém o jízdních řádech (CIS JŘ)

CIS JŘ je zákonom daný celoštátny informačný systém o cestovných poriadkoch [12].

„CIS JŘ obsahuje schválené jízdní řády linek veřejné vnitrostátní linkové dopravy (včetně městské autobusové dopravy), schválené jízdní řády linek veřejné mezinárodní linkové dopravy, které mají na území ČR zastávku pro nástup nebo výstup cestujících a schválené jízdní řády veřejné drážní osobní dopravy na dráze celostátní, regionální, tramvajové, trolejbusové, speciální a lanové provozované na území ČR." [13]

Vedením CIS JŘ je Ministerstvom dopravy Českej Republiky poverená spoločnosť *CHAPS spol s r.o.*

Obsahuje dáta vo formáte JDF, ktoré sú poskytované prostredníctvom verejne dostupného FTP serveru [14].[10] Ich automatické sťahovanie, doplnenie o chýbajúce údaje a ich konverziu do formátu GTFS zabezpečuje už spomínaný nástroj *JrUtil*. V čase písania tejto práce sú výstupy periodických konverzií dostupné na adrese.[11]

Medzi hlavné problémy tohto zdroja dát patria nepresné (resp. neúplné, s poprehadzovanými časťami) názvy zastávok a absencia súradníc [15]. V procese konverzie dát do formátu GTFS je potrebné priradiť súradnice ku zastávkam, čo nepresné názvy značne komplikujú. V prípade, že sa polohu zastávke nepodarí priradiť, niektoré spoje prechádzajú cez tzv. *Null Island*.[12] Taktiež graf zostavený z týchto dát obsahuje podozrivo málo hrán, čo môže indikovať chyby v dátach, alebo v nástroji *JrUtil* (viď porovnanie konfigurácií *full* a *cisjr* v sekcii 3.4.6.1). Z týchto dôvodov sa táto práca bude zaoberať iba oficiálnymi výstupmi v GTFS formáte.

3.1.4 Správa železnic, státní organizace (SŽ)

Živé polohy všetkých vlakov nachádzajúcich sa na území ČR, vrátane ich meškania je možné zobraziť vo webovej aplikácii *GRAPP*.[13] Klientska časť tejto aplikácie komunikuje so serverovou časťou prostredníctvom prehľadného API, z ktorého by bolo možné získať dáta o polohách vlakov pomerne priamočiaro.

O prístup k tomuto API a ku dátam vo formáte GTFS som začiatkom novembra 2023 požiadal prostredníctvom e-mailu podateľne uvedenej na stránkach SŽ. Bola mi doručená odpoveď, že mám o prístup zažiadať na inej adrese, spolu s presným popisom požadovaného rozsahu dát. Po odoslaní tejto žiadosti som však doposiaľ nedostal žiadnu odpoveď.

3.1.5 Leo Express Global a.s. (Leo Express)

Začiatkom roku 2023 bola skúmaná možnosť predaja cestovných lístkov prostredníctvom vyhľadávača *Najdispoj*. Spomedzi oslovených dopravných spoločností bol *Leo Express* jedinou, ktorá prejavila záujem o spoluprácu. Napriek tomu, že predaj cestovných lístkov je doposiaľ len v prípravnej fáze, bol v rámci zmluvy získaný prístup ku

[10] ftp://ftp.cisjr.cz/
[11] https://data.jr.ggu.cz/results/latest/
[12] https://en.wikipedia.org/wiki/Null_Island
[13] https://grapp.spravazeleznic.cz/

GTFS dátam tejto spoločnosti. Obsahujú všetky autobusové a vlakové linky daného dopravcu.

3.1.6 Integrovaný dopravní systém Jihomoravského kraje (IDS JMK)

GTFS dáta IDS JMK zverejňuje *Statutární město Brno* na portále ArcGIS Hub [16] a je možné ich stiahnuť prostredníctvom GET requestu bez nutnosti akejkoľvek autentifikácie. Sú valídne a nie je na nich nutné vykonávať žiadne úpravy. Zároveň sú k dispozícii aj dynamické dáta v dvoch verziách:

- **GTFS Realtime** – aktuálne nefunkčný zdroj
- **WebSocket Stream**[14]

3.1.7 Železnice Slovenskej republiky (ŽSR)

ŽSR zverejňujú tzv. *Grafikon vlakovej dopravy vo formáte GTFS*. Napriek tomu, že popis na portále otvorených dát *data.slovensko.sk* napovedá, že by sa malo jednať o grafikon všetkých vlakov osobnej dopravy na území SR vo formáte GTFS [17], jedná sa len o dáta spoločnosti *Železničná spoločnosť Slovensko, a.s.*. Obsahujú všetky vlakové linky danej spoločnosti, s presahom do zahraničia pri medzinárodných spojoch. Podobne ako u predchádzajúceho zdroja (IDS JMK), dáta je možné stiahnuť bez autentifikácie a sú bezprostredne použiteľné.

Dynamické dáta sú zverejnené v aplikácii podobnej *GRAPP*-u (obe aplikácie majú rovnakého prevádzkovateľa – *OLTIS Slovakia, s.r.o.* / *OLTIS Group, a.s.*) na adrese.[15] Zahrnuté sú všetky vlaky na prepravu cestujúcich, aktuálne sa nachádzajúce na území SR. Z dát je možné vyčítať polohu vlaku s presnosťou na zastávku a informácie o jeho prípadnom meškaní. Vo všeobecných podmienkach použitia je explicitne zakázané „automatické sťahovanie, prípadne ďalšie spracovanie dát so zámerom ich poskytovania tretím osobám“ [18]. Preto som podal žiadosť o povolenie k použitiu rozhrania tejto aplikácie.

Po niekoľkých týždňoch komunikácie mi bola doručená obchodná ponuka na prístup k dátam, ktorú som vzhľadom na nekomerčný charakter projektu a navrhovanú cenu neprijal.

3.1.8 Dopravný podnik Bratislava, a.s. (DPB)

Statické aj dynamické dáta DPB sú pre *Najdispoj* poskytnuté na základe zmluvy o spolupráci.[16] Statické dáta sú však už dostupné aj na portáli *Open Data Bratislava*.[17] Viac informácií k poskytovaným statickým dátam je možné nájsť v už spomenutej bakalárskej práci, v ktorej sú tieto dáta použité.

[14] https://hub.arcgis.com/datasets/mestobrno::polohy-vozidel-hromadn%C3%A9-dopravy-public-transit-positional-data/about

[15] mapa.zsr.sk

[16] https://dpb.sk/sk/dokument/zmluva-o-spolupraci-46

[17] https://opendata.bratislava.sk/dataset/category/doprava

Dynamické dáta vzhľadom na vyššiu zložitosť spracovania v bakalárskej práci neboli využité. Jedná sa o tok UDP dát, z ktorého je možné vyčítať údaje ako poloha vozidiel, smer jazdy, či rýchlosť. Meškanie nie je súčasťou týchto dát – je možné, že by sa dalo dopočítať po (zložitom) spárovaní vozidiel so statickými dátami.

3.1.9 ARRIVA Mobility Solutions, s.r.o. (AMS)

AMS zabezpečuje medzimestskú autobusovú dopravu v Bratislavskom kraji a blízkom okolí. Jej cestovné poriadky zverejňuje vo formáte GTFS spoločnosť *Bratislavská integrovaná doprava, a.s.* ako súčasť otvorených dát *Integrovaného dopravného systému v Bratislavskom kraji (IDS BK)*. Dáta sú zverejňované prostredníctvom služby *Google Drive*.

Keďže *Integrovaný dopravný systém v Bratislavskom kraji* je tvorený spoločnosťami *Dopravný podnik Bratislava, a.s.* (MHD v Bratislave), *ARRIVA Mobility Solutions, s.r.o.* (medzimestská autobusová doprava) a *Železničná spoločnosť Slovensko, a.s.* (železničná doprava), a dáta všetkých týchto spoločností sú verejne dostupné, je možné vytvoriť otvorený vyhľadávač spojov pre celý tento dopravný systém.

3.1.10 Dopravní podnik měst Liberce a Jablonce nad Nisou, a. s. (DPMLJ)

Informácie ohľadom otvorených dát DPMLJ sú dostupné na adrese.[18] GTFS dáta je možné jednoducho stiahnuť pomocou GET requestu.[19]

3.1.11 Pražská integrovaná doprava (PID)

Regionální organizátor pražské integrované dopravy, p. o. (ROPID) poskytuje pravidelne aktualizované dátové sady týkajúce sa rôznych aspektov PID.[20] Medzi ne patria aj GTFS dáta a aktuálne informácie o polohe vozidiel a dianí v dopravnom systéme vo formáte GTFS Realtime. Oba tieto zdroje dát sú najkvalitnejšie svojho druhu v rámci ČR a SR.

3.1.12 OpenStreetMap (OSM)

OpenStreetMap je bezplatná, dobrovoľníkmi tvorená mapa sveta [19], vydaná pod licenciou *OpenStreetMap License*,[21] ktorá umožňuje voľne kopírovať a distribuovať mapové dáta pre ľubovoľný účel za predpokladu, že je uvedený zdroj (OpenStreetMap a jej prispievatelia).

Na získanie OSM dát je možné použiť napríklad služby *Overpass API* alebo *Geofabrik* (viď sekcia 3.4.3.1).

[18] https://www.dpmlj.cz/opendata
[19] https://www.dpmlj.cz/gtfs.zip
[20] https://pid.cz/en/opendata/
[21] https://www.openstreetmap.org/copyright

3.2 Nástroje na prácu so zdrojovými dátami

3.2.1 Jupyter + GTFS kit

GTFS kit[22] je knižnica pre jazyk *Python*, ktorá slúži na jednoduchú analýzu GTFS dát v pamäti, bez nutnosti ich ukladania do databázy.

V kombinácii s *Jupyterom* sa jedná o vhodný nástroj na vizualizáciu a kontrolu nových zdrojov dát. V priečinku */notebooks* sa nachádza niekoľko tzv. *notebookov* využívaných na tieto účely.

3.2.2 QGIS

Použitý na vizuálnu kontrolu zdrojových dát a na vytvorenie máp v tejto diplomovej práci. Načítať dáta z GTFS je možné s pomocou pluginu *GTFS-GO*,[23] *OpenStreetMap* s pomocou *QuickOSM*.[24]

[22] https://github.com/mrcagney/gtfs_kit
[23] https://plugins.qgis.org/plugins/GTFS-GO-master/
[24] https://plugins.qgis.org/plugins/QuickOSM/

3.3 Klientska časť

Klientska časť vychádza z pôvodnej verzie vyhľadávača, keďže táto časť bola považovaná za relatívne funkčnú a použiteľnú.

Ku koncu vývoja pôvodnej klientskej časti sa začali objavovať problémy s ťažkopádnosťou pridávania novej funkcionality a neudržateľným množstvom tzv. *boilerplate* kódu. Preto bolo rozhodnuté o zásadnom prepracovaní klientskej časti aplikácie.

3.3.1 Typescript

Pôvodná klientska časť neobsahovala žiadne typové anotácie. Tento nedostatok spôsoboval časté chyby pri vývoji, ktoré sa prejavovali až pri behu aplikácie, ktorým bolo možné predísť pomocou statickej analýzy kódu. Preto bola značná časť kódu prekonvertovaná na *Typescript* a boli zavedené nové *TypeScript* triedy reprezentujúce hlavné dátové modely aplikácie (*Coords*, *Itinerary*, *Leg*, *Plan*, *Vehicle*, ...).

3.3.2 Pinia

Ďalšia veľká zmena sa týka state managementu. Pôvodná aplikácia uchovávala stav aplikácie v koreňovom komponente (*Page*) a stav (*state*) s funkciou na úpravu stavu (*updateState*) boli predávané cez *props* do vnorených komponentov – tzv. *prop drilling*.[25] To spôsobilo obrovské množstvo *boilerplate* kódu a zníženú čitateľnosť kódu.

Napriek tomu, že na vyriešenie vyššie uvedených problémov postačí aj funkcia *reactive*,[26] aktuálne odporúčanou knižnicou pre state management v aplikáciách využívajúcich *Vue 3* je *Pinia* [20]. Tá poskytuje okrem silnejších konvencií pre tímovú spoluprácu aj lepšiu integráciu s vývojárskymi nástrojmi (časová os zmien, „cestovanie v čase“, inšpekcia stavu). Novovytvorené úložiská (tzv. *store*) sú zadefinované v priečinku */client/stores* a sú rozdelené podľa logických celkov:

- **general** – stav vyhľadávacieho formulára, výsledkov vyhľadávania, ostatné zatiaľ nezatriedené stavy (*sidebarOpen*, *supportsGeolocation*).
- **locale** – jazyky, prepínanie jazykov.
- **mapObject** – vozidlá na mape.
- **persisted** – úložisko pre perzistentné dáta (obľúbené spoje, posledné hľadané, uložené itineráre).

3.3.3 Nuxt.js

Ďalšia významná časť *boilerplate* kódu bola odstránená pomocou frameworku *Nuxt.js*, konkrétne s pomocou tzv. *Auto-imports*.[27] Táto funkcionalita umožňuje automatický

[25] https://vuejs.org/guide/components/provide-inject#prop-drilling
[26] https://vuejs.org/api/reactivity-core#reactive
[27] https://nuxt.com/docs/guide/concepts/auto-imports

import komponentov, ktoré sa nachádzajú v priečinkoch *components*, *composables* a *utils*. Tým pádom bolo možné odstrániť väčšinu importov a atribútov *components* z definície komponentov (obr. 5).

```
 96  127  </template>
 97  128
 98  129  <script>
 99      - import Icon from "./Icon.vue";
100      - import Point from "./Point.vue";
101      - import Datetime from "./Datetime.vue";
102      - import Advanced from "./Advanced.vue";
103      - import Saved from "./Saved.vue";
104      - import Plan from "./Plan.vue";
105      - import Header from "./Header.vue";
106      -
107  130  import { useGeneralStore } from "@/stores/general";
108  131  import { usePersistedStore } from "@/stores/persisted";
109  132
110  133  export default {
111      -   components: {
112      -     Header,
113      -     Icon,
114      -     Point,
115      -     Datetime,
116      -     Advanced,
117      -     Saved,
118      -     Plan,
119      -   },
120  134    props: {
```

Obr. 5: Príklad zjednodušenia kódu s využitím knižnice *Nuxt.js*

3.3.4 Vite

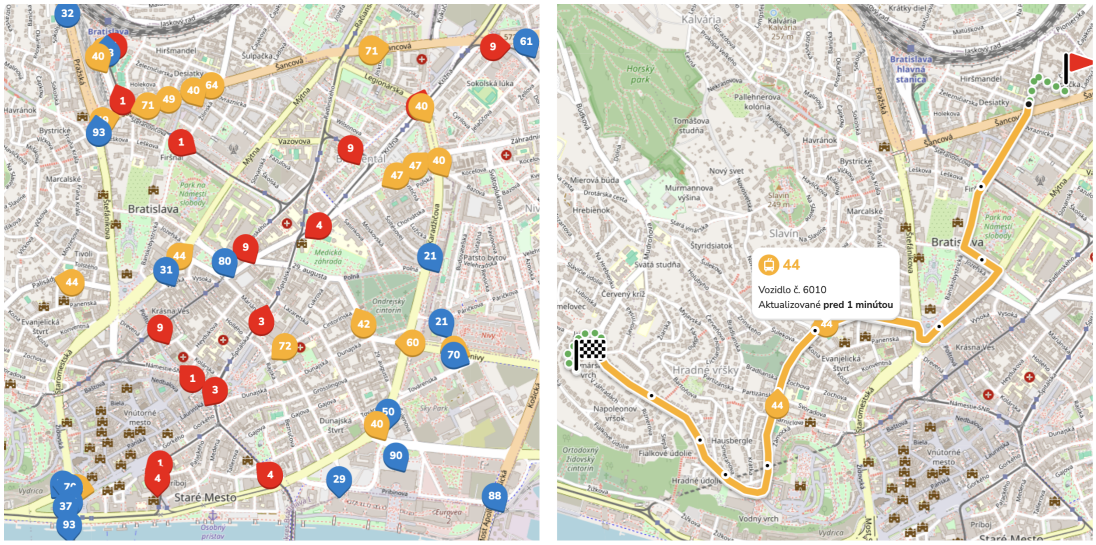
Sada vývojárskych nástrojov interne využívajúca *esbuild*[28] a *Rollup*[29] na tzv. *bundlovanie*[30] projektu [21]. Je použitá vo východzej konfigurácii frameworku *Nuxt.js*. Hlavnou výhodou *Vite* oproti *Webpacku*[31], využívanom vývojárskym prostredím *Vue CLI* (ktoré bolo použité v pôvodnej verzii a už nie je odporúčaným prostredím pre nové *Vue* projekty) je rýchlosť. V porovnaní s *Vue CLI* je *Vite* až 10× rýchlejší pri spúšťaní vývojového servera a spätná väzba pri úpravách kódu je takmer okamžitá [22].

3.3.5 Dynamické dáta

Do aplikácie bolo opätovne pridané zobrazenie živých polôh vozidiel na mape. Každé vozidlo obsahuje číslo linky, typ vozidla (autobus, trolejbus, električka), smer jazdy,

[28] https://esbuild.github.io/
[29] https://rollupjs.org/
[30] kombinovanie viacerých súborov do jedného najmä za účelom zrýchlenia načítania stránky
[31] https://webpack.js.org/

číslo vozidla a čas poslednej aktualizácie. Na mape sú zobrazované ako farebné šípky s číslom linky otočené v smere jazdy (viď obr. 6). V prípade, že je na mape zobrazený itinerár, z vozidiel sú vyfiltrované iba tie, ktoré sú súčasťou liniek v itinerári.



Obr. 6: Zobrazenie vozidiel na mape bez filtra a s filtrom

3.3.6 I18n

Internacionalizácia bola zavedená s pomocou knižnice Vue I18n.[32] Všetky textové reťazce boli preložené do angličtiny, češtiny a slovenčiny – tieto preklady sú uložené v priečinku */client/locales*. Konfigurácia podporovaných jazykov, predvoleného a záložného jazyka sa nachádza v súbore *config.ts*. Prepínač jazykov je umiestnený v pravom hornom rohu bočného panelu.

3.3.7 Progresívna Webová Aplikácia

Progresívne webové aplikácie (PWA) sú aplikácie vytvorené pomocou webových technológií. Sú teda dostupné na všetkých platformách s prehliadačom podporujúcim najnovšie štandardy. Jednou z kľúčových vlastností PWA je možnosť pridania aplikácie na domovskú obrazovku ako zdanlivo natívnu aplikáciu bez návštevy obchodu s aplikáciami.

Aby bolo možné PWA pridať na domovskú obrazovku, musia byť splnené nasledujúce kritériá [23]:

- **Web App Manifest** – odkaz na súbor *manifest.json*, ktorý obsahuje metadáta o aplikácii (napríklad názov, popis, ikony). Niektoré z týchto metadát musia byť povinne vyplnené.
- **Bezpečný kontext** – aplikácia musí byť prístupná cez HTTPS protokol.

[32] https://kazupon.github.io/vue-i18n/

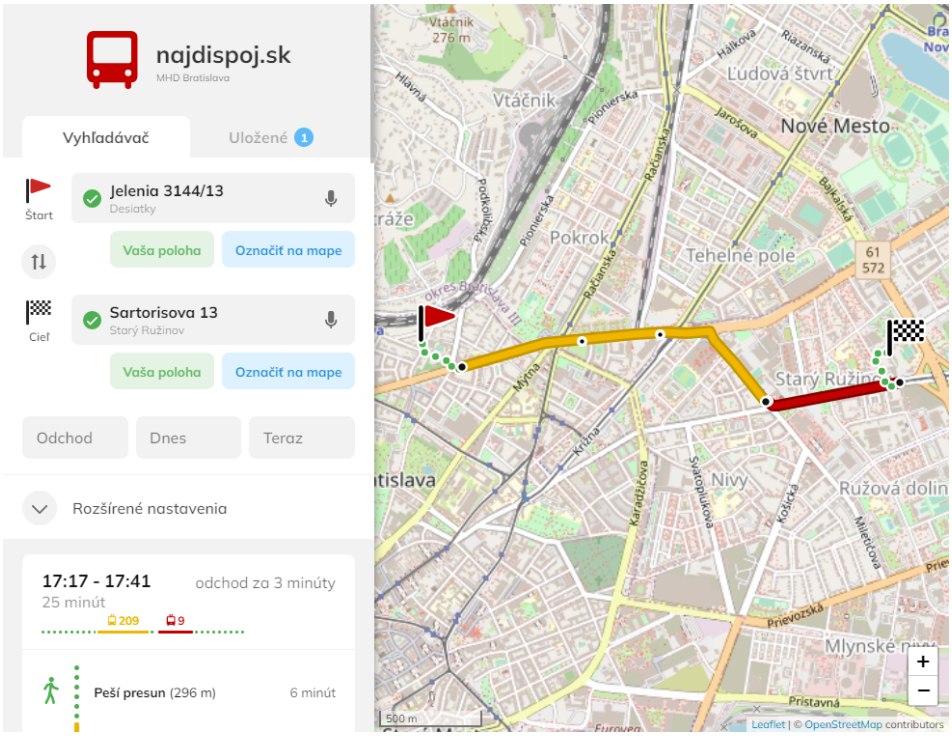
- **Service Worker** – tento skript figuruje ako proxy medzi aplikáciou, prehliadačom a sieťou. Môže napríklad umožniť používať niektoré časti aplikácie bez prístupu k internetu.

Najdispoj spĺňa všetky tieto požiadavky. Na tento účel sú použité dve knižnice (*Nuxt* moduly):

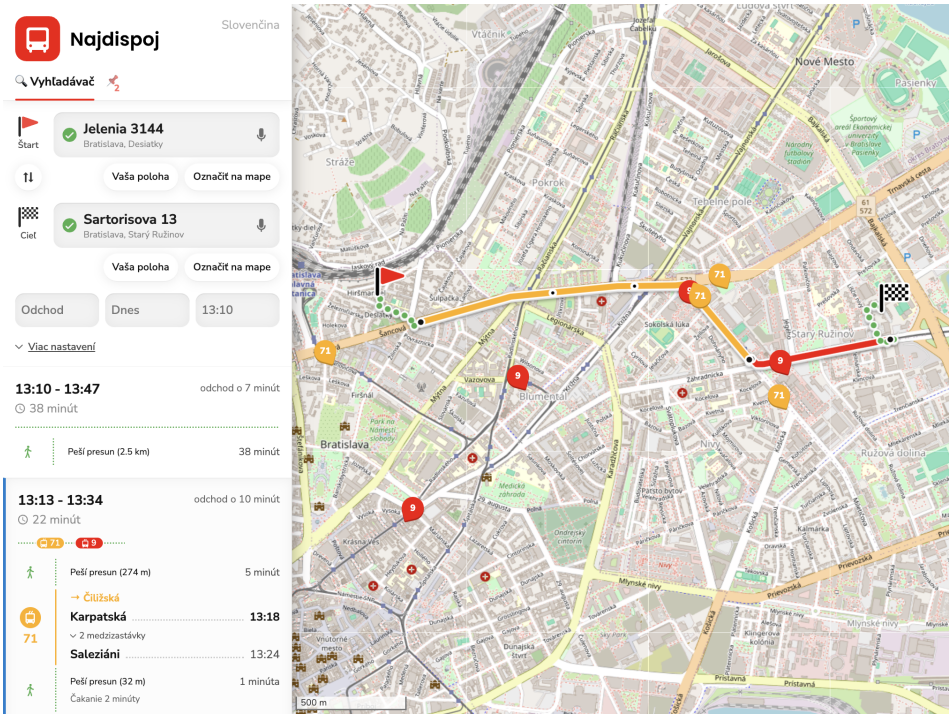
- **@vite-pwa/nuxt** – zabezpečuje generovanie manifestu, nalinkovanie service workera a ostatných potrebných súborov. Konfigurácia sa nachádza v súbore *nuxt.config.ts*.
- **@vite-pwa/assets-generator** – umožňuje automatické generovanie ikon odporúčaných rozmerov zo zdrojového svg súboru spustením jediného príkazu (*npm run generate-pwa-assets*).

3.3.8 Úpravy vzhľadu

Nová verzia klientskej časti obsahuje nové logo, obnovenú farebnú schému, zmeny súvisiace s novými zdrojmi dát (nové druhy dopravy) a zmeny súvisiace s doplnenou funkcionalitou (zobrazenie živých polôh vozidiel, prepínač jazyka). Pre porovnanie so starou časťou viď obrázky 7 a 8.



Obr. 7: Pôvodný vzhľad klientskej časti aplikácie



Obr. 8: Nový vzhľad klientskej časti aplikácie

3.4 Serverová časť

Keďže pôvodná serverová časť aplikácie bola prispôsobená konkrétnej inštancii využívajúcej len dva zdroje dát (statické dáta *Dopravného podniku Bratislava a.s.* a *OpenStreetMap* dáta z *Overpass API*), pričom pozostávala z troch súborov (*api.py*, *core.py* a *config.py*) bez akejkoľvek štruktúry, bolo potrebné ju značne prepracovať. Z tohto dôvodu sa na fungovanie novej serverovej časti pozrieme o niečo podrobnejšie.

Počas navrhovania novej serverovej časti boli zohľadnené nasledujúce požiadavky:

- **Modularita** – Existuje veľké množstvo zdrojov dát, ktoré majú často veľmi odlišné formáty a spôsoby získavania. Cieľom je poskytnúť jednoduché rozhranie pre pridanie nového zdroja dát a umožniť využívať viacero zdrojov dát rôznych typov naraz. To isté sa týka aj vyhľadávacích algoritmov, či geokódovacích služieb.
- **Možnosť zjednotiť inštancie** – V prípade, že viacero nadšencov pre dopravu získa prístup k dátam v susediacich regiónoch, jednotlivé inštancie vyhľadávača je neskôr možné triviálne spojiť do väčšej inštancie.
- **Rozdelenie zodpovednosti** – Aplikácia je rozdelená do hlavných logických celkov (služieb), ktoré sú zodpovedné za jednotlivé časti procesu vyhľadávania. V prípade získania prístupu k dátam z ďalšieho zdroja je jednoduché určiť, do ktorej oblasti patrí a vytvoriť preň tzv. poskytovateľa (*provider*).
- **Jednoduchosť nasadenia** – Vytvorenie inštancie vyhľadávača by malo byť čo najjednoduchšie a čo najmenej závislé na prostredí, čo je možné docieliť s pomocou kontajnerizácie.

3.4.1 Súborová štruktúra

Koreňový adresár serverovej časti obsahuje nasledujúce adresáre a súbory:

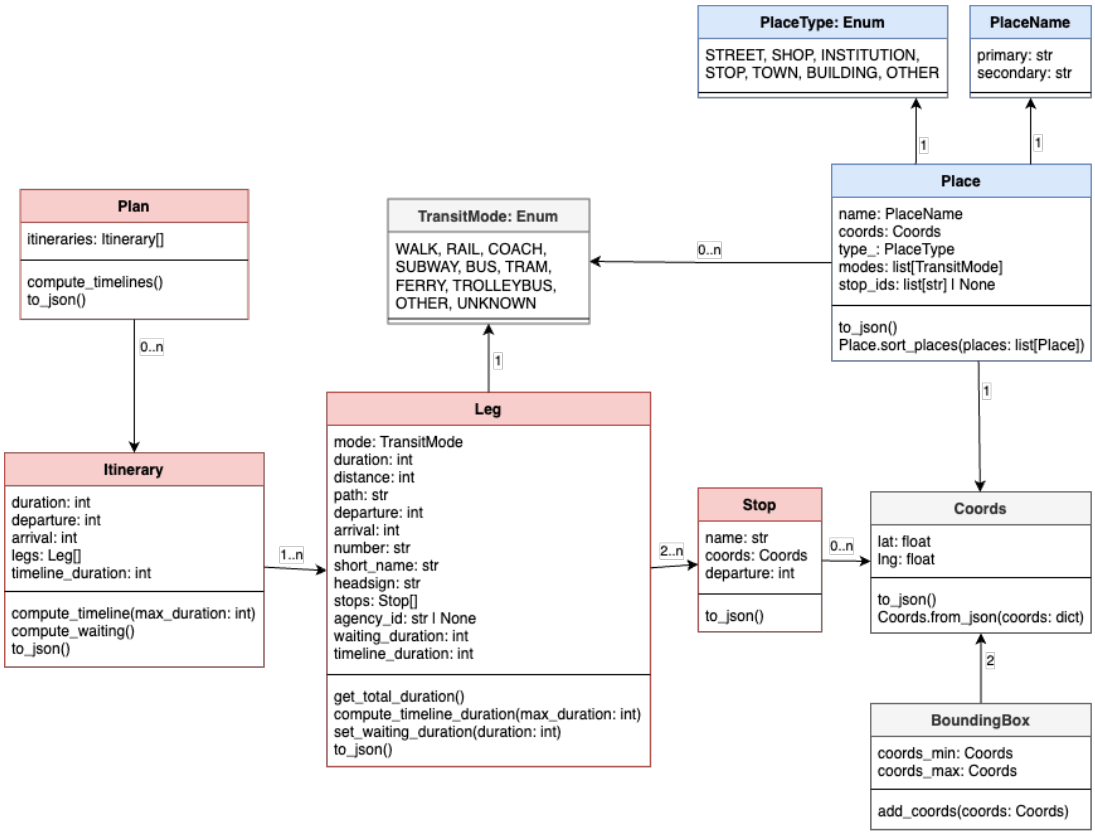
- **/models** – triedy, ktoré reprezentujú dáta a pomocné triedy (viď časť 3.4.2).
- **/services** – triedy reprezentujúce hlavné logické časti aplikácie (viď časť 3.4.3).
- **/interfaces** – rozhrania jednotlivých služieb.
- **/providers** – implementácie rozhraní.
- **/queries** – dopyty vo formáte *GraphQL*.
- **/containers** – dynamicky spúšťané *Docker* kontajnery (viď časť 3.4.5).
- **config.py** – konfiguračný súbor – ohraničenie oblasti, porty.
- **credentials.py** – konfiguračný súbor s citlivými údajmi – prihlasovacie údaje, API kľúče. Tento súbor nie je súčasťou repozitára (nachádza sa v *.gitignore*).

- **init.py** – inicializácia služieb, vytvorenie *FastAPI* aplikácie, definície endpointov a ich napojenie/presmerovanie na metódy *APIService*.
- **main.py** – časť inicializácie, ktorá sa líši od inštancie k inštancii – inicializácia zdrojov dát, ich napojenie na služby, nastavenie automatickej aktualizácie dát (tzv. *cronjobs*). Viď príklad nižšie (zdrojový kód 1).
- **run.py** – obsahuje spúšťacie skripty pre *Uvicorn*, odkazujúce na *main.py*.

```
...

@app.on_event("startup")
def init_app():
    def update_geodata():
        geodata_service.load_data()
        otp2_manager.build_street_graph()

    def update_static_data():
        gtfs_folders = static_data_service.load_data()
        otp2_manager.build_graph(gtfs_folders)

    scheduler = BackgroundScheduler()
    scheduler.add_job(update_geodata, "cron", day_of_week="sun", hour
        ="2")
    scheduler.add_job(update_static_data, "cron", hour="4")
    scheduler.start()

    update_geodata()
    update_static_data()
    otp2_manager.serve()

    nginx_manager.build_client()
```

Zdrojový kód 1: Príklad inicializácie aplikácie v súbore main.py

3.4.2 Dátový model



Obr. 9: Diagram tried serverovej časti aplikácie

V diagrame na obr. 9 sú triedy súvisiace s výsledkami vyhľadávania zvýraznené červenou farbou, triedy súvisiace so zadávaním vyhľadávania modrou farbou, triedy súvisiace so zobrazením živých dát zelenou. Vymenovávacie typy sú šedé.

3.4.2.1 Coords

Interná reprezentácia bodu na mape. Obsahuje dve číselné hodnoty – zemepisnú šírku (*lat*) a zemepisnú dĺžku (*lng*). Obsahuje *factory metódu* na vytvorenie inštancie z *JSON* objektu.

3.4.2.2 BoundingBox

Trieda reprezentujúca obdĺžnik, vymedzený dvoma bodmi na mape, ktoré predstavujú jeho severozápadný a juhovýchodný roh. V aplikácii je používaný pri práci s OSM dátami – vymedzenie oblasti pre stiahnutie dát z Overpass API, či filtrovanie dát. K dispozícii je funkcia *add_coords(coords: Coords)*, ktorá rozšíri obdĺžnik o daný bod.

3.4.2.3 Plan

Výsledky vyhľadávania. Plán obsahuje zoznam itinerárov, ktorý je v prípade neúspešného vyhľadávania prázdny. Pomocou metódy *to_json()* je možné plán previesť na objekt typu slovník, ktorý *FastAPI* automaticky prevádza na *JSON* a vráti ho ako odpoveď na HTTP požiadavku.

3.4.2.4 Itinerary

Reprezentácia itinerára vo výsledku vyhľadávania. Jeho vlastnosti sú:

- **legs** – zoznam úsekov cesty (*legs*), ktorý obsahuje aspoň jeden prvok.
- **duration** – trvanie v minútach.
- **departure** – čas odchodu zo štartu v miestnom čase.
- **arrival** – čas príchodu do cieľa v miestnom čase.
- **timeline_duration** – percentuálne trvanie itinerára vzhľadom na najdlhší itinerár v pláne. V klientskej časti sa teda jedná o šírku časovej osi, kde časová os najdlhšie trvajúceho spoja zaberá 100% šírky.

3.4.2.5 Leg

Reprezentácia úseku cesty v itinerári – môže ísť o peší presun, alebo o úsek cesty jedným spojom (vozidlom). Na základe jeho vlastností by mal byť cestujúci schopný absolvovať peší presun ku ďalšiemu úseku po navrhnutej trase, alebo identifikovať na zastávke vozidlo, ktorým sa vie dostať ku ďalšiemu úseku. Vlastnosti úseku cesty sú:

- **mode** – druh dopravy (peší presun, vlak, autobus, trolejbus, električka).
- **duration** – trvanie úseku v minútach.
- **distance** – dĺžka úseku v metroch.
- **path** – presná cesta (zoznam súradníc) úseku, zakódovaná do formátu *Encoded Polyline Algorithm Format*. Jedná sa o stratovú kompresiu zoznamu súradníc do textového reťazca (ASCII znaky) [24], za účelom drastického zníženia veľkosti dát pri prenose cez internet. Tento reťazec je v klientskej časti dekódovaný a výsledná cesta zobrazená na mape.
- **departure** – čas odchodu z prvej zastávky v miestnom čase.
- **arrival** – čas príchodu na poslednú zastávku v miestnom čase.
- **number** – pri presune vozidlom číslo linky (napríklad *4*, *X33*, *N72*. *201*). Hodnota zo stĺpca *route_short_name* z *routes.txt* (formát GTFS). Hodnota musí byť dostatočne krátka na zobrazenie v symbole vozidla na mape.

- **short_name** – dlhší variant názvu linky (napríklad *Os 3001*, *REX 1731*, *Ex 603 TATRAN*). Vo formáte GTFS zodpovedá hodnote zo stĺpca *route_long_name* v súbore *routes.txt*.
- **headsign** – nápis na tabuli vozidla, ktorým je možné identifikovať smer jeho jazdy. Zvyčajne obsahuje názov konečnej zastávky (*Nový Dvůr*, *Hlavní Nádraží*). Hodnota zo stĺpca *trip_headsign* v *trips.txt* (GTFS).
- **stops** – zoradený zoznam zastávok (objekty typu *Stop*), ktoré sú súčasťou úseku. Minimálne dva prvky.
- **agency_id** – identifikátor dopravcu, ktorý prevádzkuje vozidlo. Hodnota zo stĺpca *agency_id* v súboroch *routes.txt* a *agency.txt* (GTFS).
- **waiting_duration** – čas čakania na zastávke po pešom presune v minútach.
- **timeline_duration** – percentuálne trvanie úseku vzhľadom celkové trvanie itinerára. Spolu s *Itinerary.timeline_duration* sa používa na zobrazenie časovej osi v klientskej časti.

3.4.2.6 Stop

Reprezentácia zastávky vo výsledku vyhľadávania. Obsahuje názov zastávky, jej súradnice a čas odjazdu, špecifický pre konkrétny itinerár.

3.4.2.7 TransitMode

Vymenovávací typ reprezentujúci druhy dopravy. Obsahuje hodnoty *WALK*, *RAIL*, *COACH*, *SUBWAY*, *BUS*, *TRAM*, *FERRY*, *TROLLEYBUS*, *OTHER*, *UNKNOWN*. Implementácie *IRoutingService* mapujú GTFS hodnoty z *route_type* (súbor *routes.txt*) na tieto hodnoty.

3.4.2.8 Place

Miesto zo zoznamu v našepkávači (rozbaľovacie menu) pri textovom zadávaní štartu a cieľa. Každé miesto obsahuje:

- **name** – názov miesta, definovaný ako objekt typu *PlaceName* (viď nižšie).
- **coords** – súradnice miesta. Typ *Coords*.
- **type_** – typ miesta z vymenovávacieho typu *PlaceType* (viď nižšie). Podčiarkovník je k názvu pridaný, pretože *type* je kľúčové slovo v *Pythone*.
- **modes** – miesta vrátené triedou *OtpXapianGeocodingProvider* sú kolekcie zastávok, u ktorých tento obsahuje druhy dopravy, ktoré ich obsluhujú. Zoznam hodnôt typu *TransitMode*.

- **stop_ids** – v prípade, že sa jedná o zastávku, obsahuje zoznam GTFS identifikátorov súvisiacich zastávok (nástupíšť).

Ďalej poskytuje statickú metódu *Place.sort_places(places: list[Place])*, ktorá zoradí miesta podľa priority vo vyhľadávaní. Zoraďuje podľa hodnoty atribútu *type_* (v poradí: zastávka, inštitúcia, obchod, budova, ulica, ostatné).

3.4.2.9 PlaceName

Názov miesta, zložený z primárneho a sekundárneho názvu. Sekundárny názov môže byť názov mestskej štvrti či obce, pre rozlíšenie viacerých miest s rovnakým primárnym názvom.

3.4.2.10 PlaceType

Vymenovávací typ obsahujúci typy miest. Obsahuje hodnoty *STREET*, *SHOP*, *INSTITUTION*, *STOP*, *TOWN*, *BUILDING*, *OTHER*.

3.4.2.11 OSMFile

Pomocná trieda poskytujúca základné metódy na prácu so súbormi vo formátoch *.osm* a *.pbf*. Umožňujú jednoduchšie spracovanie dát z *OpenStreetMap* pred použitím v aplikácii. Pre svoju funkcionalitu využíva knižnicu *osmium*. Obsahuje metódy:

- **crop(bounding_box: BoundingBox)** – vyfiltruje dáta z OSM súboru na základe zadaného obdĺžnika.
- **filter(destination_osm_file: OSMFile | None)** – v súbore ponechá len záznamy, ktoré sú potrebné na vyhľadávanie spojov (zastávky, cesty, koľajnice, ostatné komunikácie). Taktiež odstráni metadáta (napríklad autor, čas poslednej zmeny). V prípade, že je špecifikovaný parameter *destination_osmfile*, výsledok sa zapíše do tohto súboru.
- **cat(osm_file: OSMFile)** – prekopíruje dáta do zadaného *osm_file*. V prípade, že má odlišnú príponu súboru, prebehne konverzia do tohto formátu.
- **merge(osm_files: list[OSMFile], destination: OSMFile)** – statická metóda, ktorá spojí viacero OSM súborov do jedného. Prípadné duplicitné záznamy sú odstránené.

3.4.2.12 GTFSFolder

Správca priečinka s GTFS dátami. Vzhľadom k tomu, že GTFS dáta z rôznych zdrojov majú rôzne neduhy (viď sekcie 3.1 a 3.4.3.3), je potrebné ich „ujednotiť“ a dostať do podoby spracovateľnej *OpenTripPlannerom*, alebo iným vyhľadávačom (viď zdrojový kód 2). Táto trieda poskytuje niekoľko metód, ktoré tento proces značne zjednodušujú:

- **load_zip(path: str)** – GTFS dáta zo zadaného *.zip* súboru extrahuje do priečinka zadaného parametrom *path* pri inicializácii inštancie triedy GTFSFolder. Vypíše validitu dát (viď *get_feed_validity()*).

- **create_feed_info(url: str)** – vytvorí *feed_info.txt* s označením dát (*self.label*) a URL zdroja dát. Tento súbor síce nie je podľa štandardu GTFS povinný [25], ale *OpenTripPlanner* ho vyžaduje.

- **repair_feed_info(url: str)** – odstráni diakritiku zo súboru *feed_info.txt* a pridá URL zdroja dát. Obe operácie sú potrebné kvôli *OpenTripPlanneru*.

- **get_feed_validity()** – pokúsi sa vydedukovať validitu dát. Najprv skontroluje stĺpce *feed_start_date* a *feed_end_date* z *feed_info.txt* (vysoko presné). Ak sú prázdne, alebo súbor neexistuje, skontroluje *calendar.txt*, kde nájde minimum stĺpca *start_date* a maximum stĺpca *end_date*. Ak neexistuje ani tento súbor, vráti minimálny a maximálny dátum z *calendar_dates.txt* (málo presné). Inak vráti dvojicu *("", "")*.

- **repair_stops()** – odstráni zo súboru *stops.txt* všetky hodnoty zo stĺpca *parent_station*. Užitočné v prípade, že dané „rodičovské zastávky" neexistujú.

- **replace_route_types(type_from: str, type_to: str)** – nahradí všetky výskyty hodnoty *type_from* v stĺpci *route_type* v súbore *routes.txt* hodnotou *type_to*. V aplikácii využívané v kombinácii s metódou *generate_shapes()*, ktorá zlyháva v prípade, že GTFS feed obsahuje linky typu 11 (trolejbus). Okrem toho *OpenTripPlanner* očakáva trolejbusy ako linky typu 800 (z *Extended GTFS Route Types* [26]), ale v poskytovaných GTFS dátach býva zvyčajne typu 11 (podľa „nerozšírenej" špecifikácie GTFS [27]).

- **replace_agency_id(old_id: str, new_id: str)** – vo všetkých súboroch, kde sa vyskytuje stĺpec *agency_id*, nahradí všetky výskyty hodnoty *old_id* hodnotou *new_id*.

- **revert_routes()** – metóda *replace_route_types* uloží pôvodný stav súboru *routes.txt* do *routes.txt.old*. Táto metóda tento súbor premenuje na *routes.txt*.

- **convert_stop_coords()** – konvertuje súradnice zo súradnicového referenčného systému EPSG:5514 (*S-JTSK / Krovak East North* [28], využívaný v Česku a na Slovensku) do systému EPSG:4326 (*WGS84 - World Geodetic System 1984*, všeobecne používaný v GPS zariadeniach [29]).

- **generate_shapes(osm_file: OSMFile)** – pomocou nástroja Pfaedle (viď sekcia 3.2) generuje súbor *shapes.txt* na základe GTFS dát a OSM dát z *osm_file*. Súbor obsahuje presné tvary trás liniek, ktoré neovplyvňujú výsledky vyhľadávania, ale zlepšujú zobrazenie trás liniek na mape.

- **zip(zip_path: str | None = None)** – vytvorí *.zip* súbor zo všetkých súborov v priečinku. V prípade, že parameter *zip_path* nie je zadaný, súbor sa uloží do *self.path*.
- **copy(path: str)** – prekopíruje dáta do zadaného priečinka a vráti novú inštanciu triedy *GTFSFolder* pre tento priečinok.
- **set_label(label: str)** – nastaví označenie dát. Používa sa v niektorých vyššie uvedených metódach.
- **get_routes()** – načíta a vráti obsah súboru *routes.txt*. Využíva sa na získanie typu vozidla po prijatí správy o živej polohe vozidla.
- **trim_empty_lines()** – odstráni prázdne riadky zo všetkých CSV súborov.
- **remove_transfers()** – odstráni súbor *transfers.txt* obsahujúci nepovinné informácie o možných prestupoch.
- **add_feed_id(feed_id: str)** – do súboru *feed_info.txt* pridá identifikátor feedu (stĺpec *feed_id*). Používa sa pri párovaní statických a dynamických dát.

```
dpb_static_data.replace_route_types("11", "3")
.generate_shapes(merged_filtered_osm)
.revert_routes()
.replace_route_types("11", "800")
.replace_agency_id("01", "DPB")
```

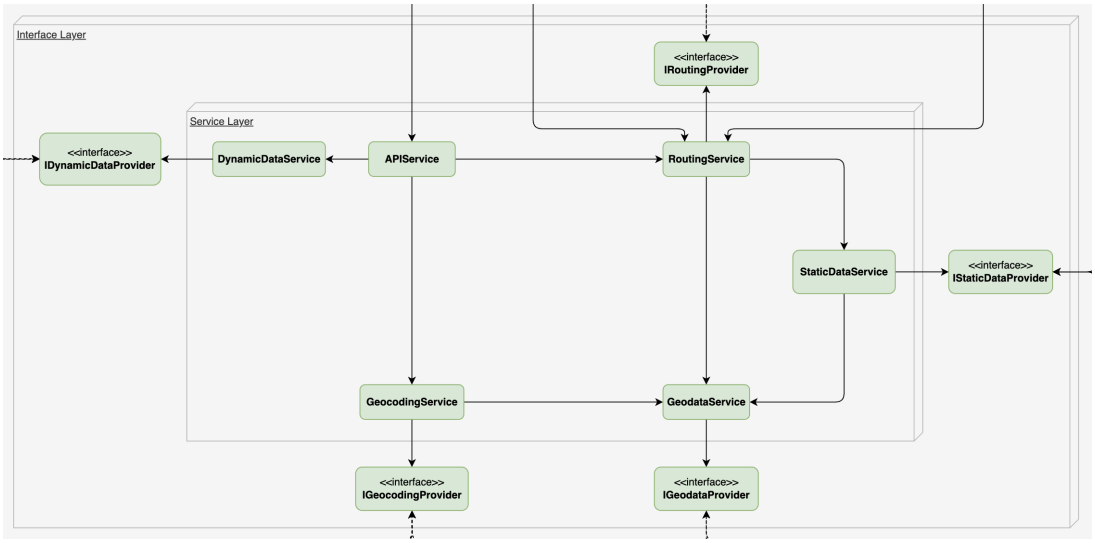
Zdrojový kód 2: Príklad spracovania GTFS dát

3.4.3 Služby

Nasledujúce podsekcie obsahujú výrezy z celkového diagramu aplikácie, ktorý je dostupný v repozitári na GitLabe[33] a v prílohe tejto práce (obr. 20).

Jadrom serverovej časti vyhľadávača je niekoľko služieb (obr. 10), ktoré zabezpečujú prístup k dátam, ich spracovanie a poskytovanie výsledkov vyhľadávania. Každá služba má definované neformálne rozhranie, ktoré je implementované jednou alebo viacerými triedami. Cieľom tejto modularity je umožniť využívať rôzne zdroje dát alebo rozhrania bez zásahu do kódu ostatných častí aplikácie.

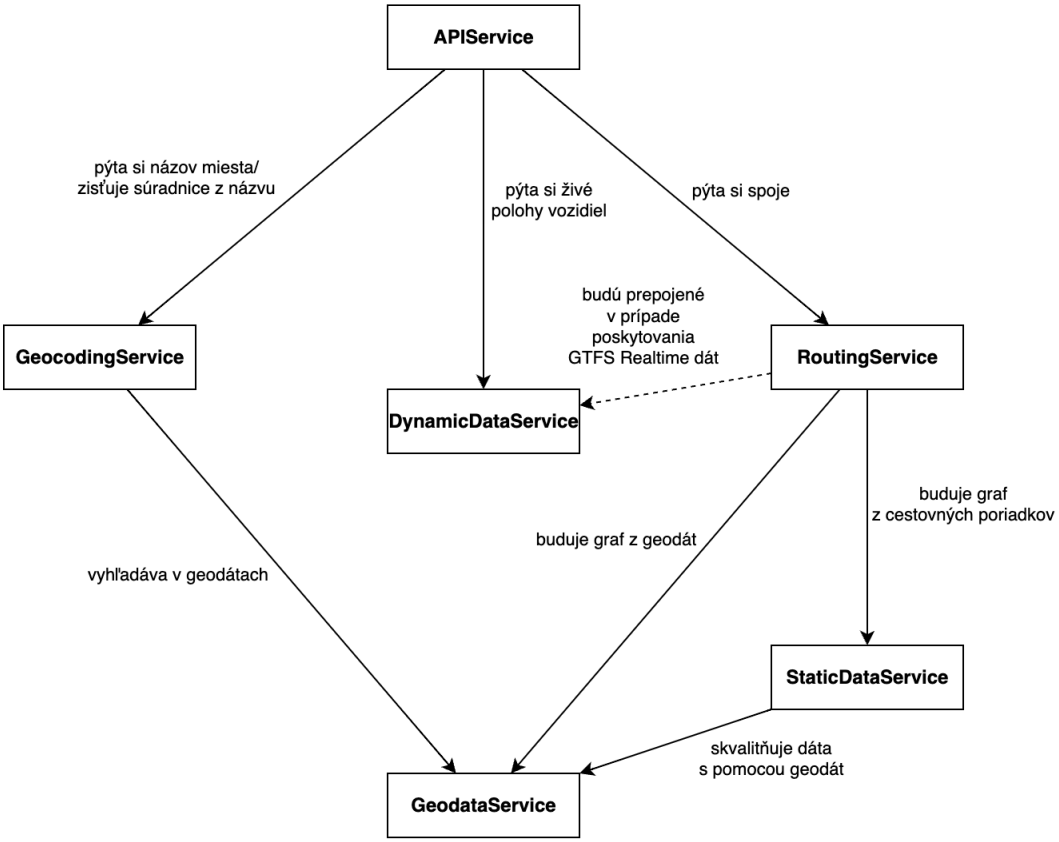
Neformálnosť je daná tým, že spektrum možných dát je veľmi široké a nie je jednoduché predpokladať, akým spôsobom môžu byť použité. Niektoré zdroje môžu obsahovať užitočnú funkcionalitu, ktorú ostatné zdroje nepodporujú. Bolo ale vyvinuté úsilie, aby každé rozhranie obsahovalo minimálne jednu metódu, ktorá je pre všetky implementácie spoločná a umožňuje aspoň základnú funkcionalitu ktorékoľvek poskytovateľa.



Obr. 10: Diagram služieb

Služby tvoria istú hierarchiu (acyklický graf), keďže niektoré služby sú závislé na výstupe iných služieb. Jednoduchým príkladom môže byť závislosť služby *RoutingService* na výstupe služby *StaticDataService* – bez cestovných poriadkov nie je možné vyhľadávať spoje. Služba *GeodataService* je naopak nezávislá na ostatných službách, a teda sa nachádza na spodku hierarchie. Táto hierarchia je znázornená na obr. 11 a podrobnejšie vysvetlená v nasledujúcich podsekciách.

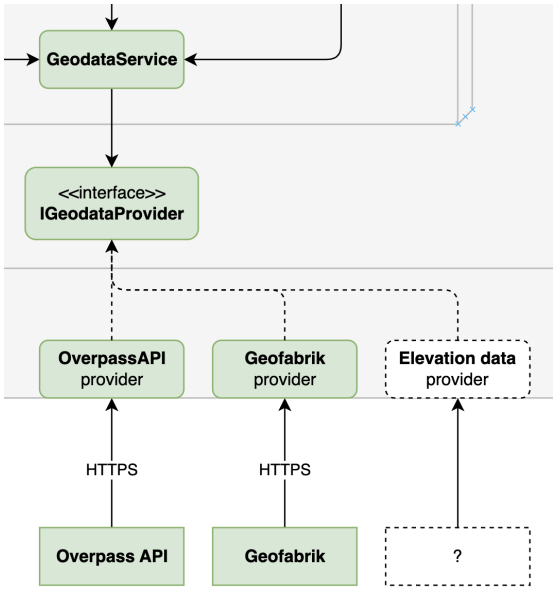
[33] https://gitlab.com/cstanislav/najdispoj/-/blob/master/static/diagram.svg



Obr. 11: Hierarchia služieb

3.4.3.1 GeodataService

Táto služba (znázornená na obr. 12) má za úlohu zaobstarať geografické dáta *OpenStreetMap* pre využitie v ostatných službách. Aktuálne sú to služby *RoutingService* (konštrukcia grafu), *StaticDataService* (zvýšenie kvality vstupných dopravných dát) a *GeocodingService*. Posledný menovaný dáta aktuálne nevyužíva, ale prepojenie existuje pre prípad, že bude potrebné spustiť vlastnú inštanciu geokódera, ktorý by ich využíval (napríklad *Photon*[34]). Neformálne rozhranie tejto služby (zdrojový kód 3) s jedinou metódou *load_data* je definované v triede *IGeodataProvider*.



Obr. 12: Diagram služby GeodataService

```
1  from server.models.osm_file import OSMFile
2
3
4  class IGeodataProvider:
5      """
6      An informal interface for geodata providers.
7      """
8
9      def load_data(self) -> OSMFile | None:
10         raise NotImplementedError()
```

Zdrojový kód 3: Rozhranie IGeodataProvider

Dané rozhranie je implementované nasledujúcimi triedami:

[34] https://photon.komoot.io/

- **GeofabrikGeodataProvider** – Táto implementácia je primárne zameraná na získanie dát zo služby *Geofabrik*, kde sa nachádzajú výstrižky z *OpenStreetMap* dát pre rôzne regióny sveta [30]. Keďže je ale parametrom konštruktora celá cesta k súboru, je možné použiť ľubovoľný zdroj dát. V aplikácii je využívaná na získanie dát pre Českú republiku a Slovensko.

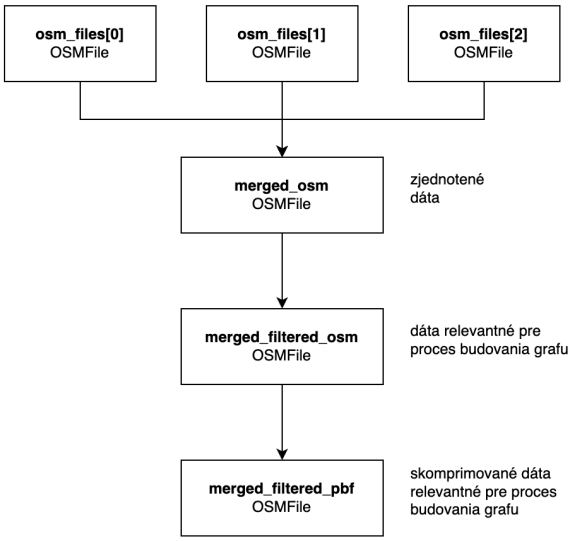
- **OverpassAPIGeodataProvider** – Umožňuje získať *OpenStreetMap* dáta pre oblasť vytýčenú dvoma bodmi na mape (objekt *BoundingBox*) zo služby *Overpass API*.[35] V prípade inštancie aplikácie pre oblasť mesta alebo kraja je to teda vhodnejšia voľba.

Keďže *OpenTripPlanner* dokáže pri konštrukcii grafu zohľadniť aj výškové dáta (formát *.tiff*/*.tif*) [31], je možné v budúcnosti implementovať službu, ktorá by zabezpečovala získanie týchto dát.

Služba *GeodataService* poskytuje dve metódy:

- **register_provider(name: str, provider: IGeodataProvider)** – pridá poskytovateľa dát do zoznamu poskytovateľov.

- **load_data(providers: list[str] = [])** – pre každého poskytovateľa dát zavolá metódu *load_data()*, výsledky zjednotí, prefiltruje a skomprimuje do formátu *.osm.pbf*. Objekty s odkazmi na výsledné súbory týchto operácií sú uchované v atribútoch (*merged_osm*, *merged_filtered_osm*, *merged_filtered_pbf*) triedy *GeodataService* pre použitie v ostatných službách. Vráti zoznam úspešne stiahnutých súborov (objekty *OSMFile*). Tento proces je znázornený na obr. 13.

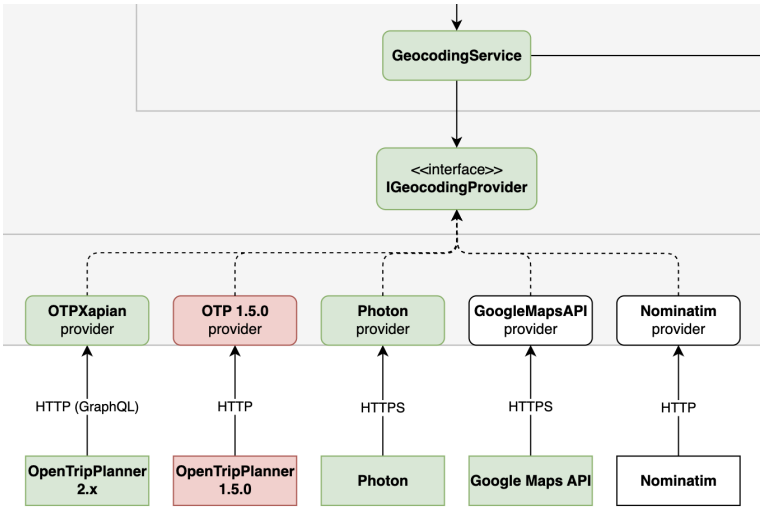


Obr. 13: Proces spracovania geografických dát

[35] https://overpass-api.de/

3.4.3.2 GeocodingService

Účelom tejto služby (znázornená na obr. 14) je poskytovať informácie o miestach na základe zadaných súradníc alebo názvu miesta. Poskytuje neformálne rozhranie IGeocodingProvider (zdrojový kód 4), ktoré definuje dve metódy: *reverse_geocode* a *autocomplete*.



Obr. 14: Diagram služby GeocodingService

```
1  from server.models.coords import Coords
2  from server.models.place import Place
3
4
5  class IGeocodingProvider:
6      """
7      An informal interface for geocoding providers - not all functions
           have to be
8      overridden by a provider class.
9      """
10
11     def reverse_geocode(self, coords: Coords) -> Place:
12         raise NotImplementedError()
13
14     def autocomplete(self, query: str) -> list[Place]:
15         raise NotImplementedError()
```

Zdrojový kód 4: Rozhranie IGeocodingProvider

- **reverse_geocode(coords: Coords)** – metóda, ktorá na základe súradníc vráti miesto (objekt typu *Place*). Je volaná po zvolení bodu kliknutím na mapu a zabezpečuje získanie názvu, oblasti a typu miesta (ikona), za účelom ich zobrazenia vo vyhľadávacom formulári a vo výsledkoch vyhľadávania.

- **autocomplete(query: str)** – metóda, ktorá na základe čiastočného názvu miesta vráti zoznam možných miest (zoznam objektov typu *Place*). Je volaná počas písania do vyhľadávacieho formulára a výsledky z nej sú zobrazené v rozbaľovacom menu pod vstupným poľom. Po zvolení jedného z výsledkov je na mape presunutá vlajka štartu, resp. cieľa na súradnice daného miesta.

Toto rozhranie je implementované troma triedami:

- **OpenTripPlannerGeocodingProvider** – čiastočná implementácia pre nástroj *OpenTripPlanner 1.5.0*. Implementuje metódu *autocomplete*, ktorá využíva GET endpoint */otp/routers/router_id/geocode*.

- **PhotonGeocodingProvider** – implementuje obe metódy rozhrania pre bežiacu inštanciu open source geokódera *Photon*. Tento geokóder, založený na fulltextovom vyhľadávači Elasticsearch, pre svoje fungovanie využíva dáta z projektu *OpenStreetMap* [32]. Trieda *PhotonGeocodingProvider* volá GET endpoint */reverse* pre metódu *reverse_geocode* a GET endpoint */api* pre metódu *autocomplete*. V predvolenej konfigurácii *Najdispoj* volá verejne dostupnú inštanciu *Photonu* na adrese,[36] ale v prípade existencie vlastnej inštancie dokáže využívať túto inštanciu úpravou parametra *address* v konštruktore triedy.

  GeoJSON je formát na výmenu geopriestorových dát založený na formáte *JSON* [33]. Keďže tento formát je výstupom z API *Photonu*, táto trieda poskytuje aj dve užitočné statické metódy pre prácu s ním:

  - **get_name_from_geojson(obj: dict)** – metóda, ktorá na základe *GeoJSON* objektu vráti názov miesta typu *PlaceName*.
  - **get_type_from_geojson(obj: dict)** – metóda, ktorá na základe *GeoJSON* objektu vráti typ miesta typu *PlaceType*.

- **OtpXapianGeocodingProvider** – trieda napísaná Davidom Koňaříkom, ktorá na základe dát z *OpenTripPlanneru 2* vytorí *Xapian* databázu, ktorá sa následne používa na full-text vyhľadávanie. Táto trieda implementuje obe metódy rozhrania. Na zber dát z OTP2 využíva jeho *GraphQL* rozhranie.

Ďalšie služby, pre ktoré by bolo možné vytvoriť implementáciu rozhrania *IGeocodingProvider*:

- **Nominatim** – open source geokóder, ktorý okrem dát z projektu *OpenStreetMap* využíva aj iné zdroje (*Wikipedia*, *US Tiger & Postcodes*, *UK Postcodes*). Na rozdiel od *Photonu* neposkytuje „search-as-you-type" funkcionalitu, ale poskytuje detailnejšie informácie o nájdených miestach [34]. V prípade implementácie rozhrania pre túto službu by preto bolo vhodné výsledky týchto dvoch zdrojov kombinovať. Podobne ako v prípade *Photonu*, je možné vytvoriť vlastnú inštanciu a využívať ju namiesto verejne dostupných inštancií, ktoré môžu byť

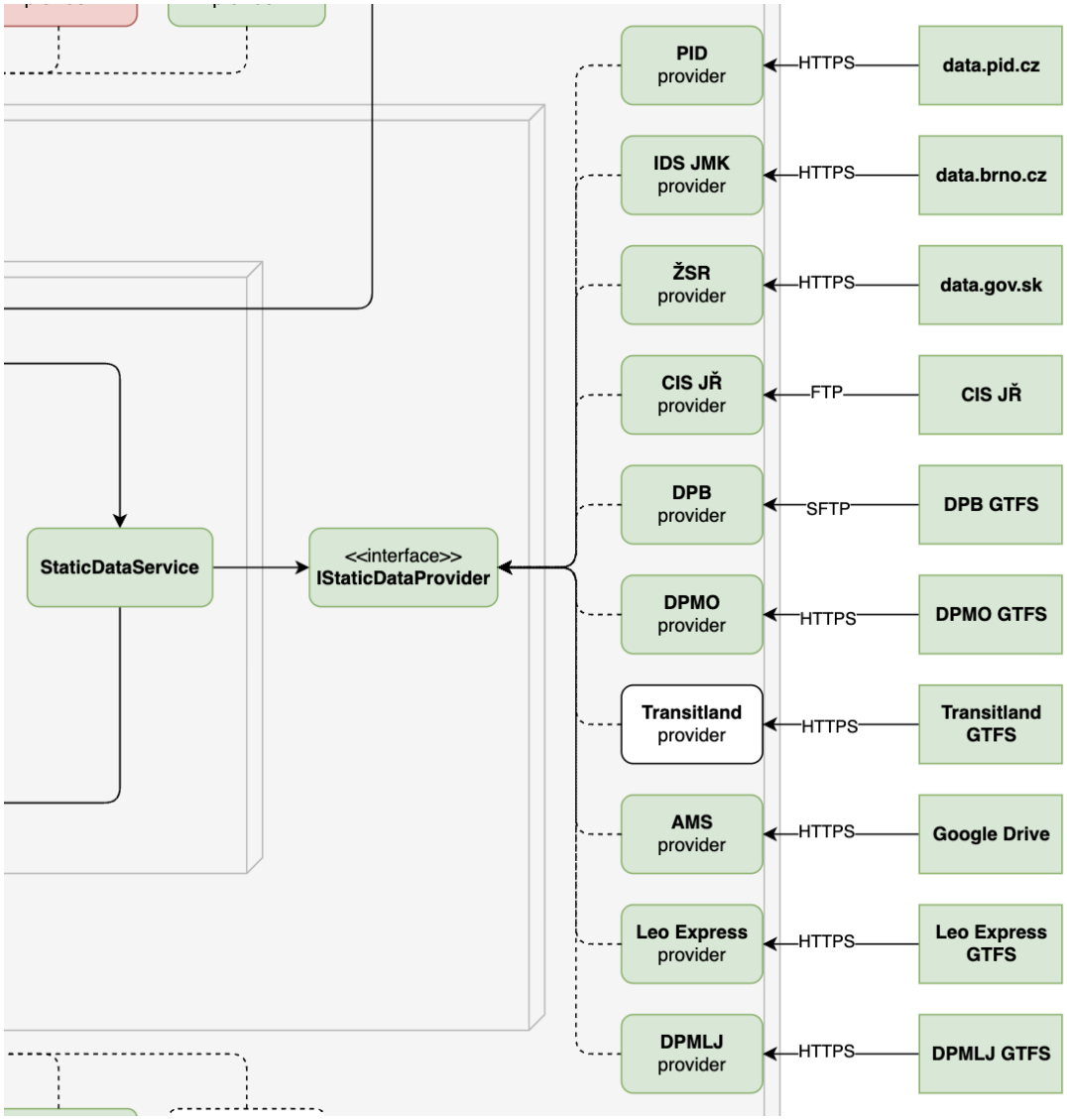
[36] https://photon.komoot.io

pomalé, alebo obsahovať obmedzenia na počet požiadaviek. Oficiálna inštancia *Nominatimu* (využívaná vyhľadávačom na stránke *openstreetmap.org*) povoľuje 1 požiadavku za sekundu, zakazuje využitie v našepkávači a vyžaduje kešovanie výsledkov [35].

- **Google Maps API** – služba poskytovaná spoločnosťou *Google*, ktorá okrem iného poskytuje geokódovanie a reverzné geokódovanie. Táto služba je platená, ale *Google* poskytuje možnosť využívať ju zadarmo do určitého limitu. *Google Maps Platform Places API* poskytuje endpoint zameraný špecificky na *autocomplete* (*Place Autocomplete* [36]) a je k dispozícii aj endpoint na *reverzné geokódovanie* (*Geocoding API* [37]). V prípade implementácie rozhrania pre túto službu by bolo vhodné využívať ju len v prípade, že iné služby nevrátia žiadne výsledky, alebo vrátia príliš málo výsledkov.

3.4.3.3 StaticDataService

Táto služba (obr. 15) sa stará o získanie a spracovanie statických dopravných dát (cestovné poriadky) do podoby, v ktorej ich dokáže skonzumovať *OpenTripPlanner* – formát GTFS. Na tento účel využíva triedu *GTFSFolder*, popísanú v časti 3.4.2 – Dátový model.



Obr. 15: Diagram služby StaticDataService

Neformálnym rozhraním služby *StaticDataService* je trieda *IStaticDataProvider* (zdrojový kód 5).

- **load_data(osm_file: OSMFile)** – metóda, ktorá na základe parametrov konštruktora daného poskytovateľa dát vytvorí objekt typu *GTFSFolder* (a priečinok, na ktorý daný objekt ukazuje) a vráti ho. V prípade zlyhania vráti *None*.

```python
from server.models.gtfs_folder import GTFSFolder
from server.models.osm_file import OSMFile


class IStaticDataProvider:
    """
    An informal interface for static data providers - not all
        functions have to be
    overridden by a provider class.
    """

    folder: GTFSFolder

    def load_data(self, osm_file: OSMFile) -> GTFSFolder | None:
        raise NotImplementedError()

    def get_folder(self) -> GTFSFolder:
        raise NotImplementedError()
```

Zdrojový kód 5: Rozhranie IStaticDataProvider

Parameter *osm_file* je potrebný pre metódu *generate_shapes()* triedy *GTFSFolder*.

- **get_folder()** – metóda, ktorá vráti objekt typu *GTFSFolder*, ktorý bol vytvorený metódou *load_data*.

Implementácie rozhrania *IStaticDataProvider*:

- **IDSJMKStaticDataProvider** – poskytuje dáta pre oblasť *Integrovaného dopravného systému Juhomoravského kraja*. Neobsahujú však súbor *shapes.txt*, obsahujúci presné tvary trás liniek, preto je nad nimi volaná metóda *generate_shapes()*.
- **ZSRStaticDataProvider** – neobsahujú súbor *shapes.txt*, ktorý je nutné vygenerovať.
- **DPMOStaticDataProvider** – dáta *Dopravného podniku mesta Olomouc*, dostupné na webe.[37] V súbore *agency.txt* neobsahujú URL poskytovateľa dát, čo je vyriešené zavolaním metódy *repair_feed_info()*.
- **DPBStaticDataProvider** – dáta *Dopravného podniku Bratislava, a.s.* sú dostupné na základe zmluvy.[38] Na základe zmluvy je umožnený prístup k pravidelne aktualizovaným dátam na serveri prostredníctvom protokolu SFTP.

[37] https://www.dpmo.cz/informace-pro-cestujici/jizdni-rady/jizdni-rady-gtfs/

[38] https://dpb.sk/sk/dokument/zmluva-o-spolupraci-c-o-229-2020

V čase zriadenia prístupu tieto dáta neboli verejne dostupné, ale v súčasnosti je ich možné získať aj prostredníctvom portálu *Open Data Bratislava*.[39]

- **AMSStaticDataProvider** – Otvorené dáta spoločnosti *ARRIVA Mobility Solutions, s.r.o.* sú dostupné v priečinku v službe *Google Drive*.[40] To spôsobuje isté ťažkosti pri automatickom spracovaní (je nutné mať *Google Service Account* a autentifikovať sa pomocou *JSON* kľúča). Dáta majú niekoľko chýb:
  - Neobsahujú súbor *feed_info.txt*, ktorý je nutné vytvoriť (s pomocou metódy *create_feed_info()*).
  - CSV súbory obsahujú na konci prázdny riadok. To spôsobuje problémy pri generovaní súboru *shapes.txt* (metóda *generate_shapes()*) nástrojom *Pfaedle*. Tento problém je vyriešený zavolaním metódy *trim_empty_lines()*.

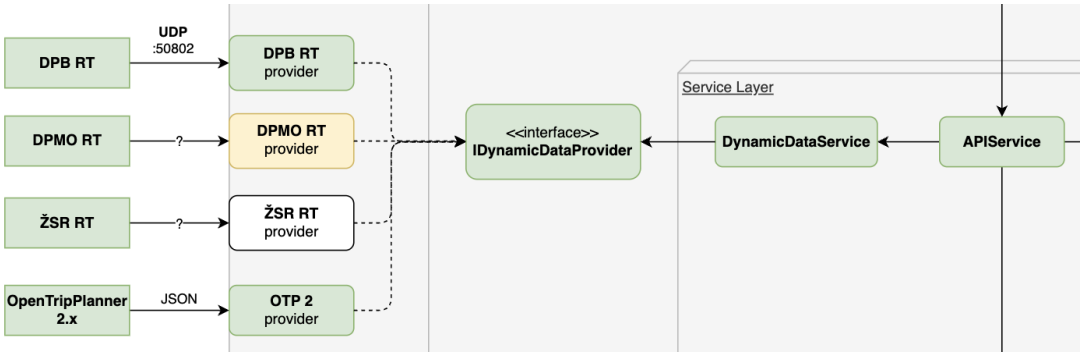
  Od sprístupnenia týchto dát je možné vytvoriť otvorený vyhľadávač spojov pre celý *Integrovaný dopravný systém v Bratislavskom kraji*.
- **LeoExpressStaticDataProvider** – dáta poskytnuté na základe zmluvy so spoločnosťou *Leo Express Global a.s.*. Neobsahujú žiadne problémy a po zavolaní metódy *generate_shapes()* sú pripravené na plnohodnotné použitie. Obsahujú všetky autobusové a vlakové linky daného dopravcu.
- **PIDStaticDataProvider** – dáta *Pražskej integrovanej dopravy* sú dostupné vo formáte GTFS na adrese.[41] Obsahujú súbor *shapes.txt*.
- **DPMLJStaticDataProvider** – *Dopravní podnik měst Liberce a Jablonce nad Nisou, a. s.* poskytuje GTFS dáta na adrese.[42] Taktiež obsahujú *shapes.txt*.
- **CISJRStaticDataProvider** – výstup z automatickej konverzie JDF dát z CIS JŘ do formátu GTFS. Problémy konverzie sú popísané v sekcii 3.1.3.

Potenciálne implementácie:

- **DPMKStaticDataProvider** – *Dopravný podnik mesta Košice, a.s.* poskytuje dáta vo formáte JDF, ktoré by teoreticky mohlo byť možné previesť do formátu GTFS. Sú dostupné na portále *Open Data Košice*,[43] avšak vzhľadom k tomu, že dáta boli naposledy aktualizované v roku 2022, nejedná sa o použiteľný zdroj dát.
- **TransitlandStaticDataProvider** – Platforma otvorených dát *Transitland* zhromažďuje GTFS, GTFS Realtime a iné dáta od poskytovateľov dopravy z celého sveta [38]. Implementácia tohto poskytovateľa by umožnila triviálne vytvorenie vyhľadávača spojov pre ktorúkoľvek oblasť sveta, kde sú dáta dostupné.

[39] https://opendata.bratislava.sk/dataset/category/doprava
[40] https://www.idsbk.sk/system/open-data/
[41] https://data.pid.cz/PID_GTFS.zip
[42] https://www.dpmlj.cz/gtfs.zip
[43] https://opendata.kosice.sk/datasets/cestovn%C3%BD-poriadok-mhd/about

3.4.3.4 DynamicDataService



Obr. 16: Diagram služby DynamicDataService

Služba zameraná na prácu so živými dátami. Časť hlavného diagramu relevantná pre túto službu je znázornená na obr. 16.

Aktuálne je využívaná len na získavanie polôh vozidiel pre ich zobrazenie na mape. V budúcnosti prichádza do úvahy práca s dátami vo formáte GTFS Realtime (polohy vozidiel, meškanie, neočakávané udalosti), avšak vzhľadom k tomu, že práca s nimi je zložitejšia a dáva zmysel v kontexte statických dát (napríklad zobrazenie meškaní, úprava vyhľadávacieho grafu), je výhodnejšie dáta napojiť priamo na *OpenTripPlanner* a získavať polohy vozidiel všeobecnejším rozhraním z neho (viď *OTP2DynamicDataProvider* nižšie).

Poskytuje rozhranie *IDynamicDataProvider* (zdrojový kód 6).

```
1  from server.models.vehicle import Vehicle
2
3
4  class IDynamicDataProvider:
5      """
6      An informal interface for dynamic data providers.
7      """
8
9      def get_vehicles(self) -> dict[str, Vehicle]:
10         raise NotImplementedError()
```

Zdrojový kód 6: Rozhranie IDynamicDataProvider

- **get_vehicles()** – metóda, ktorá vráti slovník vozidiel na základe aktuálnych dát. Kľúčom slovníka je unikátny identifikátor vozidla pre daný zdroj živých dát (napríklad číslo vozidla), hodnotou je objekt typu *Vehicle*.

Toto rozhranie je implementované dvoma triedami:

- **DPBDynamicDataProvider** – táto trieda spracováva dáta poskytované *Dopravným podnikom Bratislava*. Jedná sa o UDP tok dát, z ktorého je možné vyčítať polohu vozidiel, ich rýchlosť, kurz a smer jazdy. Keďže v dátach nie je explicitne uvedená hodnota meškania vozidla a párovať dáta z UDP toku s cestovnými poriadkami nie je triviálne, dáta aktuálne nie sú využívané na zobrazenie meškaní.

- **OTP2DynamicDataProvider** – implementácia pre *OpenTripPlanner 2*, využívajúca jeho *GraphQL* rozhranie. Umožňuje získať polohy vozidiel z viacerých zdrojov GTFS Realtime dát zároveň. V *OpenTripPlanneri* sa zdroje GTFS Realtime dát konfigurujú v súbore *router-config.json* (zdrojový kód 7). Aktuálne konfigurácia obsahuje zdroj dát od PID[44] (*Pražská integrovaná doprava*) a IDSJMK (*Integrovaný dopravný systém Juhomoravského kraja*).

[44] https://api.golemio.cz/pid/docs/openapi/#/%F0%9F%97%BA%20GTFS%20Realtime

```
{
  "updaters": [
    {
      "feedId": "PID",
      "type": "real-time-alerts",
      "url": "https://api.golemio.cz/v2/vehiclepositions/gtfsrt/
          alerts.pb",
      "frequencySec": 20
    },
    {
      "feedId": "PID",
      "type": "vehicle-positions",
      "url": "https://api.golemio.cz/v2/vehiclepositions/gtfsrt/
          vehicle_positions.pb",
      "frequencySec": 20
    },
    {
      "feedId": "PID",
      "type": "stop-time-updater",
      "url": "https://api.golemio.cz/v2/vehiclepositions/gtfsrt/
          trip_updates.pb",
      "frequencySec": 20
    },
    {
      "feedId": "IDSJMK",
      "type": "vehicle-positions",
      "url": "https://content.idsjmk.cz/kestazeni/gtfsReal.dat",
      "frequencySec": 10
    }
  ]
}
```

Zdrojový kód 7: Príklad súboru router-config.json

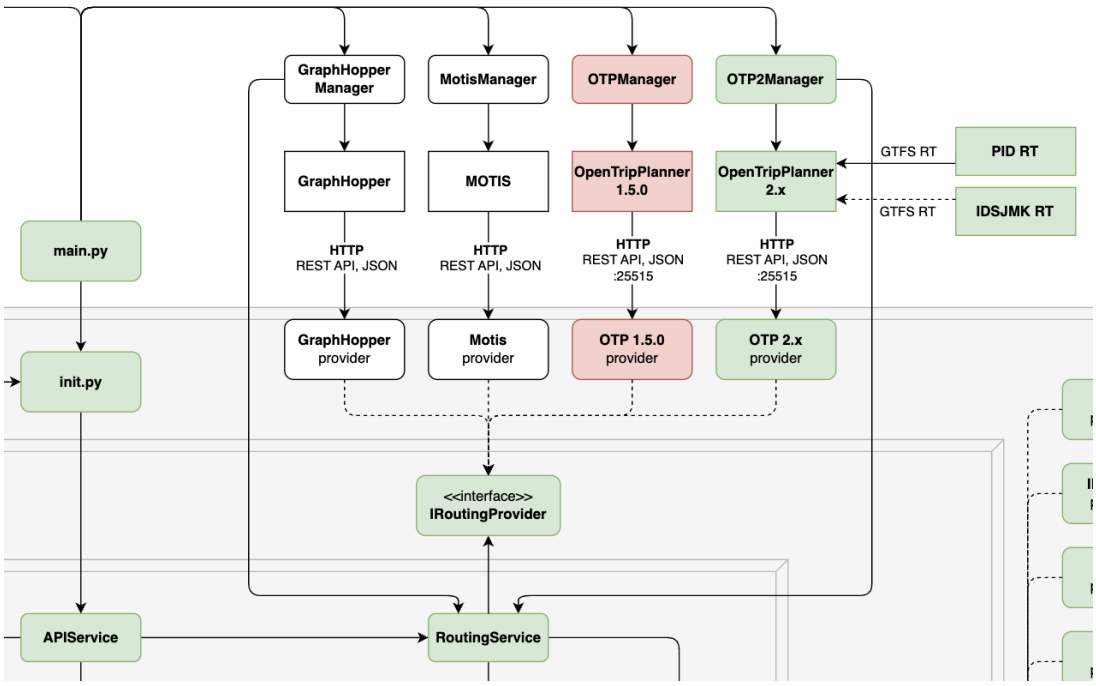
Ďalšie potenciálne implementácie rozhrania *IDynamicDataProvider* sú:

- **DPMODynamicDataProvider** – implementácia pre dáta od *Dopravného podniku mesta Olomouc*. Jedná sa o dáta z rádiovej siete, ktorá sa používa predovšetkým na zobrazenie informácií na zastávkových tabuliach. Podrobnejší popis tohto zdroja dát sa nachádza v sekcii 3.1.1.
- **ZSRDynamicDataProvider** – implementácia pre *Železnice Slovenskej republiky*. Jednalo by sa o dáta podobné tým vo webovej aplikácii na adrese.[45] Tieto dáta však nie je možné automaticky sťahovať bez povolenia od ŽSR. Pokus o získanie tohto povolenia je bližšie popísaný v sekcii 3.1.7.
- **IDSJMKDynamicDataProvider** – napriek tomu, že zdroj dát IDSJMK vo formáte GTFS Realtime je momentálne nefunkčný, existuje websocket stream, ktorý poskytuje podobné dáta. Okrem iného je z nich možné vyčítať polohy vozidiel a ich meškanie. Podrobný popis zdroja dát je dostupný na adrese.[46]

[45] https://mapa.zsr.sk/

[46] https://data.brno.cz/datasets/mestobrno::polohy-vozidel-hromadn%C3%A9-dopravy-public-transit-positional-data/about

3.4.3.5 RoutingService



Obr. 17: Diagram služby RoutingService

Služba zabezpečujúca komunikáciu s vyhľadávačmi spojov (plánovačmi cesty). Poskytuje rozhranie *IRoutingProvider* (zdrojový kód 8). Keďže má *RoutingService* prístup k *StaticDataService* aj *GeodataService*, môže byť použitá správcovskými funkciami pre vyhľadávače spojov. Relevantná časť diagramu sa nachádza na obr. 17.

```
1  from server.models.plan import Plan
2  from server.models.get_plan_params import GetPlanParams
3
4  class IRoutingProvider:
5      """
6      An informal interface for routing providers.
7      """
8
9      def get_plan(self, params: GetPlanParams) -> Plan:
10         raise NotImplementedError()
```

Zdrojový kód 8: Rozhranie IRoutingProvider

- **get_plan(params: GetPlanParams)** – metóda, ktorá na základe parametrov vráti itinerár (objekt typu *Plan*). Parametre sú získané z klientskej časti aplikácie vo formáte *JSON*. Preklad do interného formátu je zabezpečený v službe *APIService*.

Existujú dve implementácie rozhrania *IRoutingProvider*:

- **OpenTripPlannerRoutingProvider** – implementácia pre vyhľadávač *OpenTripPlanner 1.5.0*. Táto implementácia využíva HTTP GET endpoint */otp/routers/<region_id>/plan* na získanie itinerárov. Táto trieda je vzhľadom na existenciu implementácie pre *OpenTripPlanner* verzie 2 (viď nižšie) zastaraná a nie je odporúčané ju používať. Vzhľadom k tomu, že API endpoint na plánovanie cesty *OpenTripPlanner* vo verzii 1 obsahuje viac parametrov než endpoint vo verzii 2 [39], nie je vylúčené, že pre túto triedu niekto v budúcnosti nájde využitie. Preto bola zatiaľ ponechaná v kóde.

- **OpenTripPlanner2RoutingProvider** – Rozhranie pre *OpenTripPlanner 2*. Využíva HTTP POST endpoint */otp/routers/default/index/graphql*, ktorý ako vstup očakáva *GraphQL query*. *GraphQL API* je nová funkcia vo verzii 2 a umožňuje presne špecifikovať, o ktoré dáta má klient záujem, čo môže znamenať menší počet volaní API. Okrem toho umožňuje jednoduchú konštrukciu query v prehľadnom používateľskom rozhraní s výpisom dostupných polí a našepkávačom [40]. Šablóna pre tento query je umiestnená v súbore */server/queries/opentripplanner2_plan_query.graphql*. Doplnenie hodnôt do šablóny sa deje v statickej metóde *prepare_query*, vykonanie query v metóde *execute_query*.

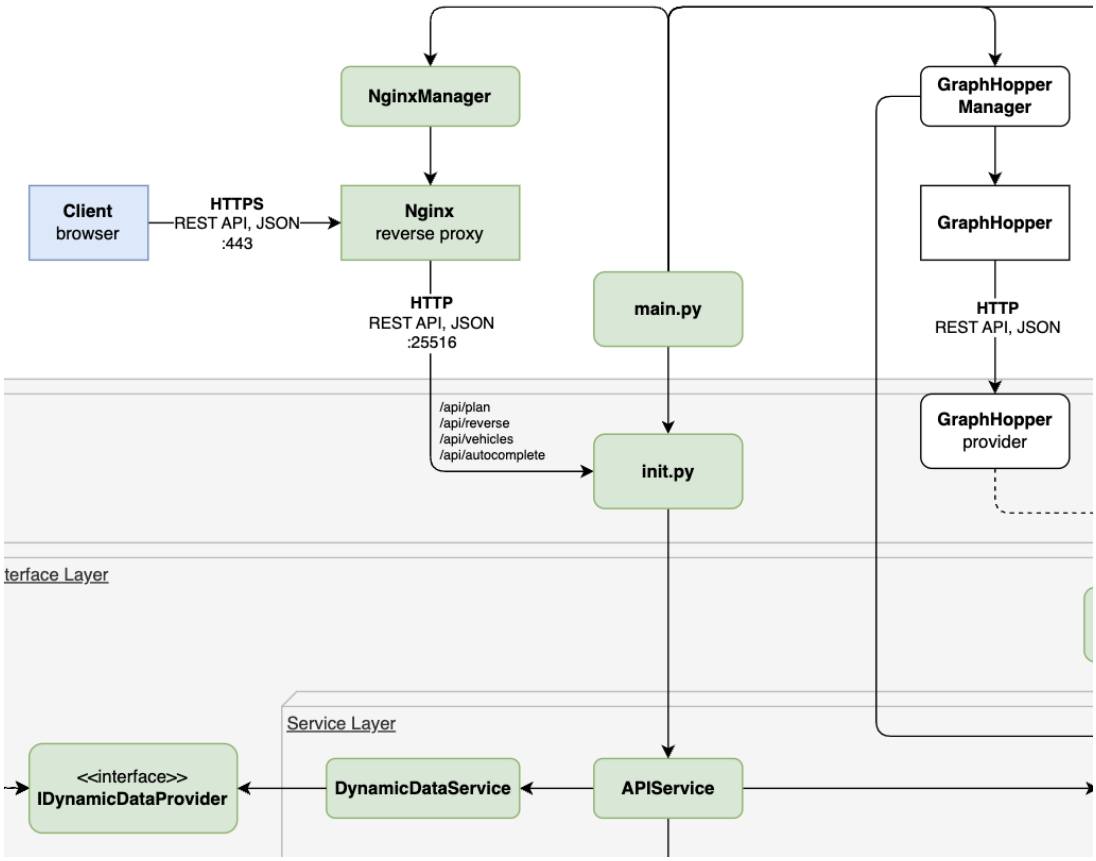
Okrem *OpenTripPlanneru* existujú aj ďalšie vyhľadávače spojov, pre ktoré by v budúcnosti mohlo stáť za pokus implementovať rozhranie *IRoutingProvider*. Za zmienku stoja najmä *GraphHopper Routing Engine* a *MOTIS*.

- **GraphHopper Routing Engine**[47] – podobne ako v prípade *OpenTripPlanneru* sa jedná o open source vyhľadávač spojov, ktorý využíva zdrojové dáta vo formátoch OSM a GTFS. Napriek tomu, že je primárne zameraný na iné druhy dopravy (automobilová, cyklistická, pešia), je možné ho využiť aj na vyhľadávanie spojov verejnej dopravy. Podporuje taktiež formát GTFS Realtime.

- **MOTIS Project** – open source projekt založený na *Technickej Univerzite Darmstadt* v spolupráci s *Deutsche Bahn* [41]. Na rozdiel od *OpenTripPlanneru* dokáže pracovať efektívnejšie s pamäťou, čo mu umožňuje bez astronomických hardvérových nárokov načítať aj väčšie oblasti (napríklad celé Nemecko).

[47] https://github.com/graphhopper/graphhopper

3.4.3.6 APIService



Obr. 18: Diagram služby APIService a jej okolia

APIService je jediná služba, ktorá neposkytuje svoje vlastné rozhranie. Sú v nej definované metódy, ktoré sú volané z klientskej časti aplikácie. Toto prepojenie je zadefinované v súbore *init.py* prostredníctvom *FastAPI*. Tieto metódy vykonajú demarshalling vstupných parametrov do internej reprezentácie a volajú príslušné metódy z *DynamicDataService*, *GeocodingService* a *RoutingService*. Obr. 18 znázorňuje vzťah klientskej časti s *APIService*.

Dostupné metódy a im príslušné API endpointy sú vypísané v tabuľke 1.

Tabuľka 1: Metódy služby APIService

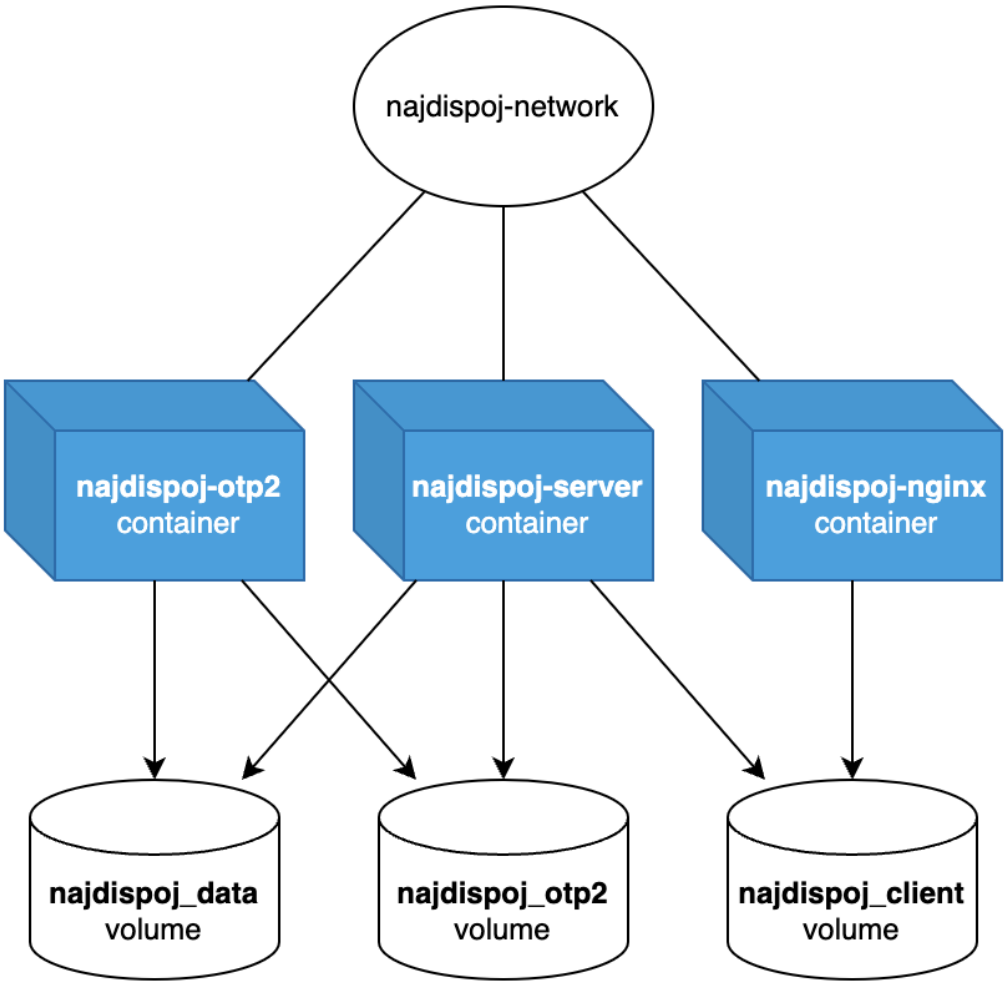
Metóda APIService	API endpoint	volaná metóda
reverse	/api/reverse	GeocodingService.reverse_geocode
autocomplete	/api/autocomplete	GeocodingService.autocomplete
plan	/api/plan	RoutingService.get_plan
vehicles	/api/vehicles	DynamicDataService.get_vehicles

3.4.4 FastAPI

FastAPI je moderný webový framework na tvorbu API napísaný v jazyku *Python*. V aplikácii *Najdispoj* je využitý na vytvorenie API endpointov pre komunikáciu s klientskou časťou aplikácie. Využíva *Uvicorn* ako *ASGI server*.[48]

3.4.5 Docker

Najdispoj je tvorený niekoľkými *Docker* kontajnermi a zväzkami (viď obr. 19). Kontajnery sú pripojené do siete *najdispoj-network*. Východzia *bridge* sieť nemôže byť využitá z dôvodu, že by kontajnery medzi sebou nemohli komunikovať podľa ich názvov.



Obr. 19: Diagram *Docker* komponentov

[48] https://asgi.readthedocs.io/en/latest/

3.4.5.1 Kontajnery

- **najdispoj-server** – hlavný kontajner, obsahujúci *FastAPI* aplikáciu a všetky nástroje na spracovanie dát (*Pfaedle*, *Osmium*). Je ho tiež možné použiť ako development container. Keďže spravuje ostatné kontajnery, je v ňom nainštalovaný *Docker* a má prístup k vonkajšiemu *Dockeru* cez *Docker daemon socket*. Toto prepojenie existuje predovšetkým kvôli uľahčeniu vývoja prostredníctvom development containeru a aplikácia by mala fungovať aj bez neho.

- **najdispoj-nginx** – kontajner slúžiaci ako reverzný proxy pre *najdispoj-server*. Obsahuje konfiguráciu pre HTTP aj HTTPS (vrátane presmerovania z www na ne-www a z HTTP na HTTPS), ako aj konfiguráciu pre tok dát UDP (pre *DPBDynamicDataProvider*). Na to využíva *Nginx*.[49]

- **najdispoj-otp2** – kontajner služby *OpenTripPlanner 2*. Jedná sa o obal nad oficiálnym kontajnerom *opentripplanner/opentripplanner*. Je spravovaný triedou *OpenTripPlanner2Manager*, ktorá poskytuje nasledujúce metódy (volané v tomto poradí):

  - **build_street_graph()** – zostaví graf bez dopravných dát (*streetGraph.obj*). Keďže geografické dáta sa menia menej často než dopravné dáta, je možné tento graf zostaviť a používať pri každom zostavovaní dopravného grafu (*graph.obj*). Pozostáva z OSM a výškových dát.
  - **build_graph()** – zostaví dopravný graf (*graph.obj*). Na tento proces potrebuje *streetGraph.obj* a GTFS dáta. Výsledný graf je možné využívať pri hľadaní spojov.
  - **serve()** – spustí server poskytujúci *GraphQL API*. Externe je dostupný na porte 25515 (východzie nastavenie, je možné zmeniť).
  - **get_instance_address()** – vráti adresu, na ktorej server je/bude dostupný pre ostatné kontajnery.

3.4.5.2 Zväzky

- **najdispoj_data** – obsahuje priečinky */gtfs* a */osm*, kde sú uložené všetky GTFS a OSM dáta.

- **najdispoj_client** – klientska časť aplikácie (priečinok */dist* po zostavení aplikácie pomocou *Vite*). Obsah tohto priečinka je servovaný kontajnerom *najdispoj-nginx*.

- **najdispoj_otp2** – využívaný kontajnerom *najdispoj-otp2* na perzistenciu dát medzi jednotlivými spusteniami. Obsahuje napríklad zostavené grafy (*streetGraph.obj*, *graph.obj*).

[49] https://nginx.org/

3.4.6 Nasadenie

Projekt obsahuje niekoľko konfigurácií (v projekte tzv. *override*) s prednastavenými zdrojmi dát a súvisiacimi nastaveniami:

- **full** – všetky verejné aj neverejné zdroje GTFS dát v ČR a SR.
- **czech_republic** – všetky verejné aj neverejné zdroje GTFS dát v ČR.
- **slovakia** – všetky verejné aj neverejné zdroje GTFS dát v SR.
- **olomouc** – dáta od DPMO, OSM dáta pre Olomouc a okolie.
- **oracle** – všetky slovenské zdroje dát a dáta od DPMO. Táto konfigurácia je využívaná na hlavnom serveri a je na hraniciach možností zdrojov daného serveru.
- **cisjr** – dáta z automatickej konverzie JDF dát z CIS JŘ.
- **integration_tests** – konfigurácia *olomouc* doplnená o integračné testy (viď sekcia 3.5).

3.4.6.1 Hardvérové nároky

Na účely zistenia hardvérových nárokov rôznych konfigurácií mi bol spoločnosťou *Váš Hosting s.r.o.* poskytnutý virtuálny server s nasledujúcimi parametrami:

- 1 vCPU (3 GHz, architektúra x86_64)
- 59 GB RAM
- 100 GB SSD
- Debian bookworm 64bit

Odskúšané boli nasledujúce konfigurácie:

- **full**
  - Trvanie inicializácie: 40 minút
  - Graf: 3,624,363 vrcholov, 9,285,845 hrán, 764 MB
  - Veľkosť *streetGraph.obj*: 626 MB
  - Maximálne využitie RAM: 30,11 GB
- **olomouc**
  - Trvanie inicializácie: 3 minúty
  - Graf: 75,489 vrcholov, 197,540 hrán, 16 MB

  - Veľkosť *streetGraph.obj*: 13 MB
  - Maximálne využitie RAM: 4,11 GB

- **cisjr**
  - Trvanie inicializácie: 1 hodina
  - Graf: 2,454,320 vrcholov, 6,284,929 hrán, 1.1 GB
  - Veľkosť *streetGraph.obj*: 415 MB
  - Maximálne využitie RAM: 48,95 GB

3.4.6.2 Servery

Aplikácia beží na dvoch serveroch:

- **Hicoria proxy** (1 vCPU, 1 GB RAM, 50 GB SSD, Debian) – pôvodne hlavný server, aktuálne zredukovaný na minimálnu konfiguráciu na hostingu *Hicoria*.[50] Je naň nasmerovaný UDP tok dynamických dát od *Dopravného podniku Bratislava*. Ten ich preposiela na *Oracle Cloud*. Zároveň sa tento server využíva aj na prístup k statickým dátam tohto dopravného podniku a dátam *Železničnej spoločnosti Slovensko* (cez SSH tunel), keďže servery, na ktorých sa nachádzajú, nie sú prístupné z hlavného serveru. V budúcnosti by dávalo zmysel nasadiť na tento server nástroj na automatické spracovanie dynamických dát na formát GTFS Realtime.

- **Oracle Cloud** (4 OCPU, 24 GB RAM, 50 GB + 150 GB SSD, Ubuntu) – hlavný server, na ktorom beží aplikácia. Server sa nachádza v Nemecku a jedná sa o VPS poskytovaný bezplatne spoločnosťou *Oracle* v rámci *Oracle Cloud Free Tier*[51] na *Arm Compute Instance* (*VM.Standard.A1.Flex shape*) s pripojeným *Block Volume Storage* o veľkosti 150 GB. Proces automatického vytvorenia infraštruktúry by v budúcnosti mohol byť riešený prostredníctvom nástroja *Terraform*.[52] Obsahuje všetky slovenské zdroje dát a dáta od DPMO.

Univerzitou poskytnutý server (1 vCPU, 25 GB RAM, 60 GB SSD, Debian) bol istý čas využívaný na vývoj, avšak vzhľadom na zložitejšiu správu (prístup len cez SSH bez monitoringu) a nižšiu spoľahlivosť bol nahradený serverom *Oracle Cloud*.

3.4.6.3 systemd

Automatické spustenie po štarte systému a reštart po páde aplikácie sú zabezpečené správcom systému a služieb *systemd*. Konfigurácia služby sa nachádza v súbore */etc/systemd/system/najdispoj.service* (zdrojový kód 9).

[50] https://hicoria.com/
[51] https://www.oracle.com/cloud/free/
[52] https://registry.terraform.io/providers/oracle/oci/latest/docs

```
[Unit]
Description=Najdispoj
After=network.target
[Service]
User=najdispoj
Restart=on-failure
RestartSec=30
ExecStart=/home/najdispoj/najdispoj/run_service.sh
[Install]
WantedBy=multi-user.target
```

Zdrojový kód 9: Konfigurácia služby *najdispoj.service*

3.4.6.4 Certifikát

Aby bola aplikácia dostupná cez protokol HTTPS, je využitý certifikát od certifikačnej autority *Let's Encrypt*[53]. Na jeho vytvorenie a obnovenie je použitý nástroj *Certbot*. Podrobnejší postup je popísaný v súbore *README.md* v koreňovom priečinku projektu.

3.4.6.5 Rozdiel medzi AArch64 a x86_64

Vzhľadom k tomu, že procesor hlavného serveru (*Oracle Cloud*) je založený na architektúre *AArch64*, je možné, že niektoré nástroje, napísané pre architektúru *x86_64*, na ňom nebudú fungovať správne. Jedným z nich je *Pfaedle*, využívaný na generovanie súboru *shapes.txt* pre GTFS dáta (viď sekcia 3.4.2.12 – GTFSFolder).

Aby sme docielili správne fungovanie programu na tejto architektúre, je nutné pred jeho zostavením zmeniť niektoré riadky kódu. Zostavenie je súčasťou *Dockerfile* a preto musí byť zmena vykonaná v ňom, v prípade, že je argument *AARCH64* nastavený na *true* (viď zdrojový kód 10). Tieto zmeny sa týkajú dátových typov (rozdiely v znamienkach).

3.5 Testovanie

3.5.1 Unit testy

Isté časti aplikácie sú pokryté unit testami. Tieto sú umiestnené v priečinku */server/tests* a využívajú knižnicu *unittest* zo štandardnej knižnice jazyka *Python*. Testy sa spúšťajú príkazom *python -m unittest*.

3.5.2 Integračné testy

Integračné testy majú podobu samostatnej konfigurácie aplikácie s názvom *integration_tests*. Testy sú umiestnené v podpriečinku */server/integration_tests*.

[53] https://letsencrypt.org/

```
1 # Make Pfaedle work on AArch64 architecture
2 RUN if [ "${AARCH64}" = "true" ]; then \
3     sed -i 's/char c;/int c;/g' ${PFAEDLE_PATH}/src/pfaedle/config/
        ConfigReader.cpp; \
4     sed -i 's/static_cast<int/static_cast<unsigned int/g'
        ${PFAEDLE_PATH}/src/cppgtfs/src/ad/util/CsvParser.cpp && \
5     sed -i 's/static_cast<signed char/static_cast<unsigned char/g'
        ${PFAEDLE_PATH}/src/cppgtfs/src/ad/util/CsvParser.cpp && \
6     sed -i 's/-17/0xEF/g' ${PFAEDLE_PATH}/src/cppgtfs/src/ad/util/
        CsvParser.cpp && \
7     sed -i 's/-69/0xBB/g' ${PFAEDLE_PATH}/src/cppgtfs/src/ad/util/
        CsvParser.cpp && \
8     sed -i 's/-65/0xBF/g' ${PFAEDLE_PATH}/src/cppgtfs/src/ad/util/
        CsvParser.cpp; \
9     fi
```

Zdrojový kód 10: Zmena v *Dockerfile* pre AArch64

Pri spustení aplikácie s touto konfiguráciou prebehnú všetky kroky bežnej konfigurácie a na záver sa spustia integračné testy:

- vytvorenie hlavného *Docker* kontajnera,
- registrácia poskytovateľov dát/služieb,
- vytvorenie a spustenie *Nginx reverse proxy*,
- stiahnutie geografických dát pre Olomouc a okolie,
- zostavenie *streetGraph.obj* pomocou nástroja *OpenTripPlanner 2*,
- stiahnutie dopravných dát DPMO,
- zostavenie *graph.obj* pomocou nástroja *OpenTripPlanner 2*,
- spustenie integračných testov.

Integračné testy využívajú knižnicu *pytest*. Jedná sa o niekoľko volaní API endpointov (cez *Nginx*) a kontrolu výstupu. Keďže konkrétne výstupy sa v závislosti na dátach môžu líšiť, je kontrolovaná predovšetkým štruktúra odpovedí. Očakáva sa napríklad, že spoj z hlavnej stanice do centra mesta bude nájdený vždy, pričom prvá a posledná časť sú pešie presuny, avšak konkrétne časy, či čísla liniek môžu byť rôzne.

3.6 Zapojenie komunity

Od predstavenia projektu *Najdispoj* na fóre *platforma.slovensko.digital*[54] sa v menšej či väčšej miere do propagácie a vývoja projektu zapojilo niekoľko ľudí (resp. subjektov), ktorí prispeli podporou, radami, nápadmi, či dokonca vylepšeniami kódu. Nižšie je uvedených niekoľko najvýznamnejších:

3.6.1 2023 Open Data Maturity Report

Hodnotenie *Open Data Maturity (ODM)* je každoročná štúdia skúmajúca dosiahnutý progres európskych štátov v oblasti otvorených dát. Jednotlivé štáty hodnotí v štyroch kategóriách: *Policy*, *Portal*, *Impact* a *Quality* [42].

Dotazník za Slovensko vypĺňa *Ministerstvo investícií, regionálneho rozvoja a informatizácie SR* a v hodnotení pre rok 2023[55] v ňom bol zahrnutý aj *Najdispoj*. V kategórii *Impact - Environmental impact* získal pre Slovensko 15 bodov a v kategórii *Impact - Economic impact* sa spolupodieľal na 20 bodoch. Celkovo Slovensko získalo 2324 bodov a umiestnilo sa na 10. mieste [43].

3.6.2 Článok na Alvaria.sk

Občianske združenie *Alvaria*, zaoberajúce sa mimo iného aj problematikou otvorených dát, zverejnilo 10. novembra 2023 článok o projekte *Najdispoj*.[56] Obsahuje stručný popis funkcionality aplikácie, zdrojových dát a problémov s ich spracovaním.

3.6.3 Nová funkcionalita, neoficiálna inštancia

David Koňařík, autor už spomínaného nástroja *JrUtil* prispel pokročilejším vyhľadávaním zastávok pomocou knižnice *Xapian* (*OtpXapianGeocodingProvider*), pridaním podpory nových druhov dopravy (metro, trajekt) a drobným refactoringom istých častí kódu.[57]

Okrem toho prevádzkuje vlastnú inštanciu vyhľadávača,[58] kde zhromažďuje primárne české zdroje dát.

[54] https://platforma.slovensko.digital/

[55] https://data.europa.eu/en/publications/open-data-maturity/2023

[56] https://www.alvaria.sk/najdispoj-sk-uzitocny-vyhladavac-spojov-verejnej-dopravy/

[57] https://gitlab.com/cstanislav/najdispoj/-/merge_requests/4

[58] https://ns.jr.ggu.cz/

Záver

V rámci tejto diplomovej práce bol vyhľadávač spojov *Najdispoj* rozšírený o nové zdroje dát a funkcionalitu. Klientska časť aplikácie sa dočkala niekoľkých vylepšení, ako napríklad zobrazenie polohy vozidiel v reálnom čase, či internacionalizáciu. Zároveň boli odstránené neduhy, ktoré brzdili jej ďalší rozvoj. Serverová časť bola vytvorená nanovo s dôrazom na vyššiu modularitu, rozšíriteľnosť a testovateľnosť. Boli pridané nové zdroje dát, predovšetkým statické dáta olomouckej mestskej hromadnej dopravy, ale aj prakticky všetky ostatné verejne dostupné zdroje dopravných dát v Českej a Slovenskej republike. Spôsob vytvorenia vlastnej inštancie bol zjednodušený s pomocou *Docker* kontajnerov. Napokon bola výsledná aplikácia nasadená na server.[59]

V budúcnosti sa rozvoj aplikácie môže uberať viacerými smermi. Je možné pokračovať aktuálnym smerom, a teda pridávať ďalšie zdroje dát z verejne dostupných zdrojov, ale aj na základe zmlúv. Ďalším smerom môže byť pridávanie funkcionality do klientskej časti aplikácie, ako napríklad zobrazenie informácií o zastávkach, či linkách. Pokus o odstraňovanie nedostatkov z automaticky skonvertovaných dát CIS JŘ by mohol znamenať vytvorenie spoľahlivého a kompletne open source vyhľadávača spojov pre celé územie Českej republiky. V neposlednom rade by mohlo byť zaujímavé použiť iné vyhľadávače spojov, ako napríklad *GraphHopper* alebo *MOTIS*, či pokúsiť sa o vytvorenie vlastného vyhľadávača, ktorý by bol optimalizovaný na miestne pomery napríklad tým, že by nevyžadoval konverziu dát do formátu GTFS.

[59] https://najdispoj.sk/

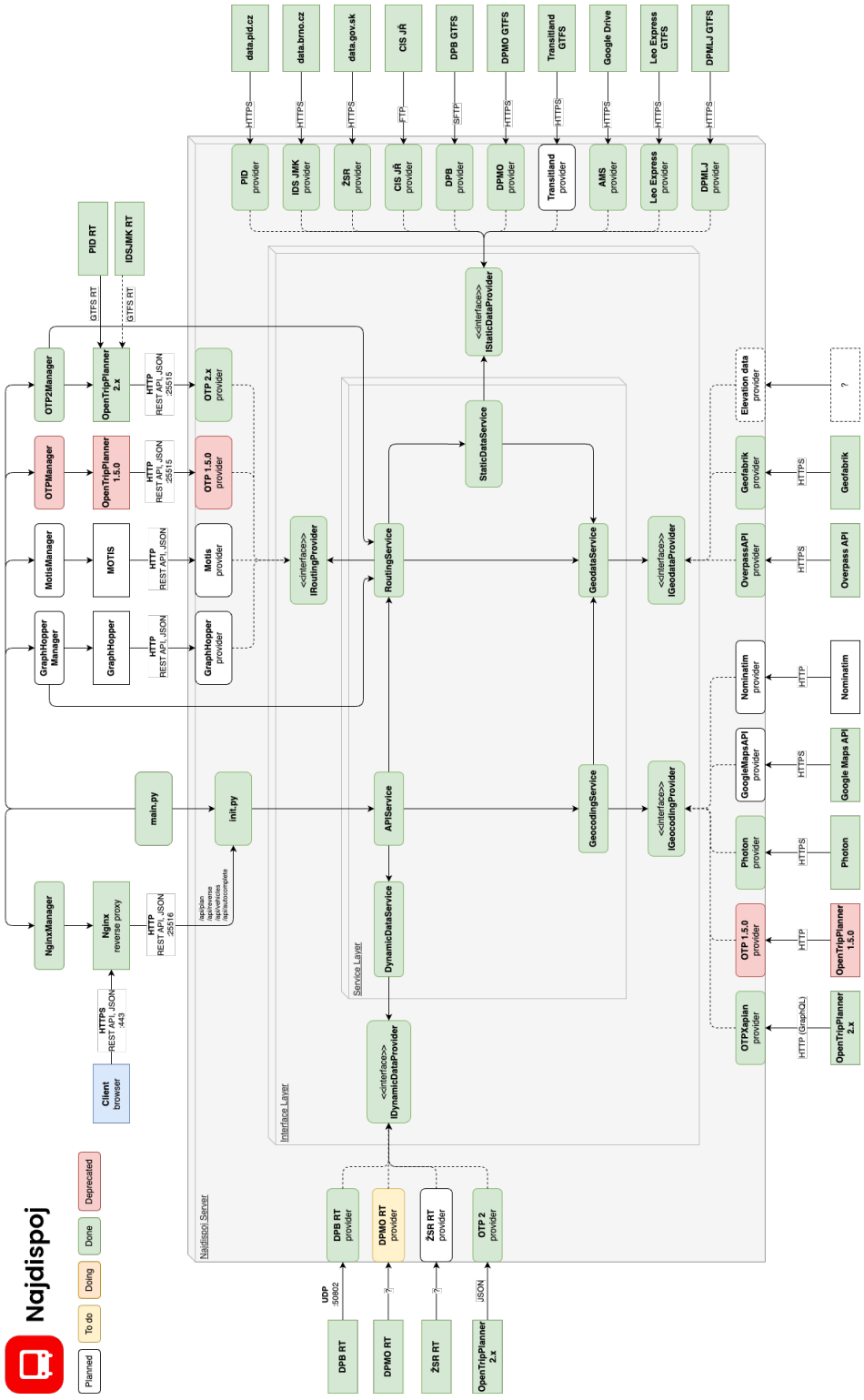
Conclusions

As part of this thesis, *Najdispoj* journey planner was enhanced by adding new data sources and functionality. The client side of the application received several improvements, such as displaying the location of vehicles in real time, or internationalization. At the same time, ailments that hindered its further development were removed. The server part was created from scratch with an emphasis on higher modularity, expandability and testability. New data sources were added, primarily static data of public transport in Olomouc, but also practically all other publicly available sources of transit data in the Czech Republic and Slovakia. Creation of a custom instance has been simplified with the help of *Docker* containers. Finally, the resulting application was deployed on the server.[60]

In the future, the development of the application can go in several directions. It is possible to continue in the current direction, by adding additional data sources from publicly available sources, but also on the basis of contracts. Another direction can be adding functionality to the client part of the application, such as displaying information about stops or lines. An attempt to remove deficiencies from the automatically converted CIS JŘ data could mean the creation of a reliable and completely open source journey planner for the entire territory of the Czech Republic. Last but not least, it could be interesting to use other search engines, such as *GraphHopper* or *MOTIS*, or try to create a custom search engine, which would be optimized for local conditions, e.g. by not requiring data conversion to GTFS format.

[60] https://najdispoj.sk/

A Diagram serverovej časti aplikácie



Obr. 20: Diagram serverovej časti aplikácie

B Obsah elektronických dat

`text/`
: Adresár s textom práce vo formáte PDF, vytvorený s použitím záväzného štýlu KI PřF UP v Olomouci pre záverečné práce, vrátane všetkých príloh, a všetky súbory potrebné pre bezproblémové vytvorenie PDF dokumentu textu.

`najdispoj.zip`
: Kompletný zdrojový kód aplikácie *Najdispoj* vo formáte ZIP archívu. Obsahuje súbor *README.md* s inštrukciami na spustenie aplikácie. Dostupné aj na odkaze.[61]

[61] https://gitlab.com/cstanislav/najdispoj

Literatúra

[1] *GTFS Static Overview | Static Transit | Google for Developers*. [online]. [cit. 2024-1-25]. Dostupné z www: ⟨https://developers.google.com/transit/gtfs⟩.

[2] Goldstein, Brett; Dyson, Lauren; Nemani, Abhi. *Beyond Transparency: Open Data and the Future of Civic Innovation*. San Francisco, CA: Code for America, 2013. Available also from WWW: ⟨https://beyondtransparency.org/pdf/BeyondTransparency.pdf⟩. ISBN 978-0615889085.

[3] *NeTEx | Network Timetable Exchange*. [online]. [cit. 2024-1-25]. Dostupné z www: ⟨https://netex-cen.eu/⟩.

[4] *How does NeTEx compare with GTFS? | NeTEx*. [online]. [cit. 2024-1-25]. Dostupné z www: ⟨https://netex-cen.eu/faq/how-does-netex-compare-with-gtfs/⟩.

[5] *Metodický pokyn č. 5 k organizaci celostátního informačního systému o jízdních řádech*. [online]. [cit. 2024-3-10]. Dostupné z www: ⟨https://www.mdcr.cz/getattachment/Dokumenty/Verejna-doprava/Jizdni-rady,-kalendare-pro-jizdni-rady,-metodi-(1)/Jizdni-rady-verejne-dopravy/metodicky-pokyn-cis-5.pdf.aspx⟩.

[6] Masopust, Jan. Automatické zpracování českých jízdních řádů autobusů. In. *Automatické zpracování českých jízdních řádů autobusů*. 2020. Available also from WWW: ⟨https://gisak.vsb.cz/GIS_Ostrava/GIS_Ova_2020/sbornik/papers/gis20205de22dca7ab01.pdf⟩.

[7] *GTFS Realtime Overview | Realtime Transit | Google for Developers*. [online]. [cit. 2024-3-10]. Dostupné z www: ⟨https://developers.google.com/transit/gtfs-realtime⟩.

[8] *Overview | Protocol Buffers Documentation*. [online]. [cit. 2024-3-10]. Dostupné z www: ⟨https://protobuf.dev/overview/⟩.

[9] *Nash Equilibrium, Pareto Optimality and Public Goods with Two Agents*. [online]. [cit. 2024-3-23]. Dostupné z www: ⟨https://eml.berkeley.edu/~webfac/saez/e131_s04/publicgoods1.pdf⟩.

[10] *Car or Public Transport—Two Worlds*. [online]. [cit. 2024-3-23]. Dostupné z www: ⟨https://ad-publications.cs.uni-freiburg.de/EfficientAlgorithms_Car_Bast_2009.pdf⟩.

[11] Delling, Daniel; Pajor, Thomas; Werneck, Renato. Round-Based Public Transit Routing. *Transportation Science*. 2012, č. 49. Available also from WWW: ⟨http://dx.doi.org/10.1287/trsc.2014.0534⟩.

[12] *CIS JŘ: Minulost, současnost a budoucnost jízdních řádů*. [online]. [cit. 2024-2-2]. Dostupné z www: ⟨https://dvdkon.ggu.cz/articles/openalt-2023-cisjr/slides/#3.0⟩.

[13] *CHAPS - Produkty - CIS JŘ*. [online]. [cit. 2024-3-9]. Dostupné z www: ⟨https://chaps.cz/cs/products/CIS⟩.

[14] *CHAPS - Produkty - CIS JŘ*. [online]. [cit. 2024-3-9]. Dostupné z www: ⟨https://web.archive.org/web/20240119115945/https://chaps.cz/cs/products/CIS⟩.

[15] *CIS JŘ: Minulost, současnost a budoucnost jízdních řádů*. [online]. [cit. 2024-3-19]. Dostupné z www: ⟨https://dvdkon.ggu.cz/articles/openalt-2023-cisjr/slides/#45.0⟩.

[16] *Jízdní řád IDS JMK ve formátu GTFS | GTFS timetable data | ArcGIS Hub*. [online]. [cit. 2024-2-26]. Dostupné z www: ⟨https://hub.arcgis.com/datasets/379d2e9a7907460c8ca7fda1f3e84328/about⟩.

[17] *Národný katalóg otvorených dát - Grafikon vlakovej dopravy vo formáte GTFS*. [online]. [cit. 2024-2-26]. Dostupné z www: ⟨https://data.slovensko.sk/datasety/ca4cb74c-7192-4198-b074-34acd9d295e7⟩.

[18] *Mapa*. [online]. [cit. 2024-3-10]. Tlačidlo „Všeobecné podmienky použitia“ v pravom dolnom rohu. Dostupné z www: ⟨https://mapa.zsr.sk/index.aspx⟩.

[19] *About OpenStreetMap – OpenStreetMap Wiki*. [online]. [cit. 2024-3-21]. Dostupné z www: ⟨https://wiki.openstreetmap.org/wiki/About_OpenStreetMap⟩.

[20] *State Management | Vue.js*. [online]. [cit. 2024-3-3]. Dostupné z www: ⟨https://vuejs.org/guide/scaling-up/state-management.html⟩.

[21] *Why Vite | Vite*. [online]. [cit. 2024-3-1]. Dostupné z www: ⟨https://vitejs.dev/guide/why.html⟩.

[22] Kelly, Daniel. *How to Migrate from Vue CLI to Vite - Vue School Articles* [online]. [cit. 2024-3-1]. Dostupné z www: ⟨https://vueschool.io/articles/vuejs-tutorials/how-to-migrate-from-vue-cli-to-vite/⟩.

[23] *Making PWAs installable - Progressive web apps | MDN*. [online]. [cit. 2024-3-17]. Dostupné z www: ⟨https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Guides/Making_PWAs_installable⟩.

[24] *Encoded Polyline Algorithm Format | Google Maps Platform | Google for Developers*. [online]. [cit. 2024-2-11]. Dostupné z www: ⟨https://developers.google.com/maps/documentation/utilities/polylinealgorithm⟩.

[25] *Reference | Static Transit | Google for Developers*. [online]. [cit. 2024-2-11]. Dostupné z www: ⟨https://developers.google.com/transit/gtfs/reference#feed_infotxt⟩.

[26] *Extended GTFS Route Types | Static Transit | Google for Developers*. [online]. [cit. 2024-2-11]. Dostupné z www: ⟨https://developers.google.com/transit/gtfs/reference/extended-route-types⟩.

[27] *Reference | Static Transit | Google for Developers*. [online]. [cit. 2024-2-11]. Dostupné z www: ⟨https://developers.google.com/transit/gtfs/reference#routestxt⟩.

[28] *S-JTSK / Krovak East North - SJTSK - EPSG:5514*. [online]. [cit. 2024-2-11]. Dostupné z www: ⟨https://epsg.io/5514⟩.

[29] *WGS 84 - WGS84 - World Geodetic System 1984, used in GPS - EPSG:4326*. [online]. [cit. 2024-2-11]. Dostupné z www: ⟨https://epsg.io/4326⟩.

[30] *Geofabrik Download Server*. [online]. [cit. 2024-2-23]. Dostupné z www: ⟨https://download.geofabrik.de/⟩.

[31] *Build - OpenTripPlanner 2*. [online]. [cit. 2024-2-23]. Dostupné z www: ⟨https://docs.opentripplanner.org/en/v2.4.0/BuildConfiguration/#elevation-data⟩.

[32] *komoot/photon: an open source geocoder for openstreetmap data*. [online]. [cit. 2024-2-10]. Dostupné z www: ⟨https://github.com/komoot/photon⟩.

[33] Butler, Howard; Daly, Martin; Doyle, Allan a kol. *The GeoJSON Format*. 2016. 28 str. Request for Comments. RFC 7946⟨https://www.rfc-editor.org/info/rfc7946⟩.

[34] *Nominatim vs Photon geocoder | Geoapify*. [online]. 2019 [cit. 2024-2-10]. Dostupné z www: ⟨https://www.geoapify.com/nominatim-vs-photon-geocoder⟩.

[35] *Nominatim Usage Policy (aka Geocoding Policy)*. [online]. [cit. 2024-2-10]. Dostupné z www: ⟨https://operations.osmfoundation.org/policies/nominatim/⟩.

[36] *Place Autocomplete | Places API | Google for Developers*. [online]. [cit. 2024-2-10]. Dostupné z www: ⟨https://developers.google.com/maps/documentation/places/web-service/autocomplete⟩.

[37] *Reverse geocoding (address lookup) request and response | Geocoding API | Google for Developers*. [online]. [cit. 2024-2-10]. Dostupné z www: ⟨https://developers.google.com/maps/documentation/geocoding/requests-reverse-geocoding⟩.

[38] *Transitland • Welcome*. [online]. [cit. 2024-2-26]. Dostupné z www: ⟨https://www.transit.land/⟩.

[39] *Comparing OTP2 to OTP1 - OpenTripPlanner 2*. [online]. [cit. 2024-2-22]. Dostupné z www: ⟨https://docs.opentripplanner.org/en/dev-2.x/Version-Comparison/#commentary-on-otp1-features-removed-from-otp2⟩.

[40] *GraphQL | A query language for your API*. [online]. [cit. 2024-2-22]. Dostupné z www: ⟨https://graphql.org/⟩.

[41] *MOTIS Project | MOTIS is an intermodal travel information system that supports GTFS Real-Time. It is open source and provides a HTTP JSON API, web and Android app.* [online]. [cit. 2024-2-26]. Dostupné z www: ⟨https://motis-project.de/⟩.

[42] *Open Data Maturity in Europe | EU Data Portal | data.europa.eu.* [online]. [cit. 2024-3-21]. Dostupné z www: ⟨https://data.europa.eu/en/publications/open-data-maturity⟩.

[43] *ODM2023_results_overview.xlsx.* [online]. [cit. 2024-3-22]. Dostupné z www: ⟨https://data.europa.eu/sites/default/files/ODM2023_results_overview.xlsx⟩.