

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-16607-104434

**VIZUALIZÁCIA POHYBU POUŽÍVATEĽOV  
DIPLOMOVÁ PRÁCA**

**2024**

**Bc. Richard Búri**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-16607-104434

**VIZUALIZÁCIA POHYBU POUŽÍVATEĽOV  
DIPLOMOVÁ PRÁCA**

Študijný program: Aplikovaná informatika  
Názov študijného odboru: Informatika  
Školiace pracovisko: Ústav informatiky a matematiky  
Vedúci záverečnej práce: Ing. Maroš Čavojský, PhD.

**Bratislava 2024**

**Bc. Richard Búri**



## ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Richard Búri**

ID študenta: 104434

Študijný program: aplikovaná informatika

Študijný odbor: informatika

Vedúci práce: Ing. Maroš Čavojský, PhD.

Vedúci pracoviska: doc. Ing. Milan Vojvoda, PhD.

Miesto vypracovania: Ústav informatiky a matematiky

Názov práce: **Vizualizácia pohybu používateľov**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je navrhnuť a implementovať algoritmus, ktorý bude schopný vizualizovať väčšie datasety GPS trajektórií prehľadne (používateľ bude schopný zrakovo zistíť základné pohybové vzory z trajektórií). Výstup algoritmu by mal byť zobrazený interaktívnym spôsobom (priblížovanie, oddaľovanie) v internetovom prehliadači.

Úlohy:

1. Naštudujte si aktuálne spôsoby pri zobrazovaní väčšieho množstva GPS trajektórií v spoločnej oblasti.
2. Navrhnite algoritmus pre analýzu vstupných dát, pre ich následnú vhodnú vizualizáciu v internetovom prehliadači.
3. Implementujte navrhnutý algoritmus ako aj zobrazovanie výstupu algoritmu interaktívnym spôsobom v internetovom prehliadači
4. Otestujte a vyhodnoťte implementovaný algoritmus a zobrazovanie v internetovom prehliadači pre rôzne GPS datasety.

Zoznam odbornej literatúry:

1. ČAVOJSKÝ, Maroš; DROZDA, Martin. Search by pattern in GPS trajectories. In: TAHERI, Javid; VILLARI, Massimo; GALLETTA, Antonino. *Mobile Computing, Applications, and Services*. Cham: Springer, 2023, s. 117–132. ISBN 978-3-031-31890-0.
2. ČAVOJSKÝ, Maroš; DROZDA, Martin; BALOGH, Zoltán. Analysis and experimental evaluation of the Needleman-Wunsch algorithm for trajectory comparison. *Expert Systems with Applications*, 165. s. 2021.
3. Dokumentácia GraphHopper – Map matching pre GPS trajektórie, URL: <https://docs.graphhopper.com/>

# SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program: Aplikovaná informatika

Autor: Bc. Richard Bíri

Diplomová práca: Vizualizácia pohybu používateľov

Vedúci záverečnej práce: Ing. Maroš Čavojský, PhD.

Miesto a rok predloženia práce: Bratislava 2024

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean et est a dui semper facilisis. Pellentesque placerat elit a nunc. Nullam tortor odio, rutrum quis, egestas ut, posuere sed, felis. Vestibulum placerat feugiat nisl. Suspendisse lacinia, odio non feugiat vestibulum, sem erat blandit metus, ac nonummy magna odio pharetra felis. Vivamus vehicula velit non metus faucibus auctor. Nam sed augue. Donec orci. Cras eget diam et dolor dapibus sollicitudin. In lacinia, tellus vitae laoreet ultrices, lectus ligula dictum dui, eget condimentum velit dui vitae ante. Nulla nonummy augue nec pede. Pellentesque ut nulla. Donec at libero. Pellentesque at nisl ac nisi fermentum viverra. Praesent odio. Phasellus tincidunt diam ut ipsum. Donec eget est. A skúška mäkčeňov a dĺžnov.

Kľúčové slová: webová aplikácia, javascript, map-matching, trasy

# ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Bc. Richard Búri
Master's thesis:	Visualization of user's movement
Supervisor:	Ing. Maroš Čavojský, PhD.
Place and year of submission:	Bratislava 2024

On the other hand, we denounce with righteous indignation and dislike men who are so beguiled and demoralized by the charms of pleasure of the moment, so blinded by desire, that they cannot foresee the pain and trouble that are bound to ensue; and equal blame belongs to those who fail in their duty through weakness of will, which is the same as saying through shrinking from toil and pain. These cases are perfectly simple and easy to distinguish. In a free hour, when our power of choice is untrammelled and when nothing prevents our being able to do what we like best, every pleasure is to be welcomed and every pain avoided. But in certain circumstances and owing to the claims of duty or the obligations of business it will frequently occur that pleasures have to be repudiated and annoyances accepted. The wise man therefore always holds in these matters to this principle of selection: he rejects pleasures to secure other greater pleasures, or else he endures pains to avoid worse pains.

Keywords: web application, javascript, map-matching, routes

## Vyhľásenie autora

Richard Búri čestne vyhlasujem, že som diplomovú prácu Vizualizácia pohybu používateľov vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci. Vedúcim mojej diplomovej práce bol Ing. Maroš Čavojský, PhD.

Bratislava, dňa 10.5.2024

## **Podakovanie**

Chcem sa podakovať vedúcemu záverečnej práce, ktorým bol Ing. Maroš Čavojský, PhD., za odborné vedenie, rady a pripomienky, ktoré mi pomohli pri vypracovaní tejto diplomovej práce.

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Problematika vývoja webových aplikácií</b>	<b>2</b>
1.1 Webové aplikácie a ich vývoj . . . . .	2
1.2 Vývojový proces v 8 krokoch: . . . . .	2
1.3 Výber technológií . . . . .	3
<b>2 Podobné aplikácie</b>	<b>5</b>
2.1 Mapbox . . . . .	5
2.2 GPS Visualizer . . . . .	6
2.3 Routiak . . . . .	7
<b>3 Použité technológie</b>	<b>10</b>
3.1 Docker . . . . .	10
3.2 Valhalla api . . . . .	10
3.2.1 Odin (generovanie trasy, rozpoznávanie manévrov) . . . . .	10
3.2.2 Thor (trasovacie algoritmy) . . . . .	11
3.2.3 Sif (cena pre hrany a prechody) . . . . .	11
3.2.4 Loki (priradenie lokácia grafovým vrcholom) . . . . .	11
3.2.5 Mjolnir (trasovací graf a generovanie dlaždíc) . . . . .	11
3.2.6 Meili (map matching) . . . . .	12
3.3 Geojson.io . . . . .	12
3.4 Kepler.gl . . . . .	13
3.5 Express (Node.js) . . . . .	14
3.6 Visual Studio Code . . . . .	15
3.7 MySQL . . . . .	16
3.8 Leaflet . . . . .	17
<b>4 Návrh riešenia</b>	<b>18</b>
4.1 Špecifikácia požiadaviek . . . . .	18
4.1.1 Funkcionálne požiadavky . . . . .	18
4.1.2 Nefunkcionálne požiadavky . . . . .	19
4.1.3 Diagramy . . . . .	20
4.2 Ukladanie údajov . . . . .	22
4.3 Pripínanie trás k cestnej sieti . . . . .	24

<b>5</b>	<b>Implementácia</b>	<b>28</b>
5.1	Pripínanie trasy k cestnej sieti . . . . .	28
<b>6</b>	<b>Zhodnotenie</b>	<b>32</b>
	<b>Záver</b>	<b>33</b>
	<b>Zoznam použitej literatúry</b>	<b>34</b>

# Zoznam obrázkov a tabuliek

Obrázok 1	Porovnanie Mapbox a Valhalla map-matching . . . . .	6
Obrázok 2	GPS Visualizer prostredie . . . . .	7
Obrázok 3	Zobrazenie väčšieho množstva trás na rovnakom mieste . . . . .	8
Obrázok 4	Zobrazenie väčšieho množstva trás upravených map-matchom na rovnakom mieste . . . . .	9
Obrázok 5	Prostredie Geojson.io . . . . .	13
Obrázok 6	Prostredie Kepler.gl . . . . .	14
Obrázok 7	Prostredie Visual Studio Code . . . . .	16
Obrázok 8	Diagram prípadov použitia . . . . .	20
Obrázok 9	Sekvenčný diagram procesu nahrania súborov na server . . . . .	21
Obrázok 10	Sekvenčný diagram procesu zobrazenia trás na mape . . . . .	21
Obrázok 11	Sekvenčný diagram procesu zobrazenia chybových súborov . . . . .	22
Obrázok 12	Štruktúry priečinkov <i>uploads</i> a <i>routes</i> . . . . .	23
Obrázok 13	Trasa pripnutá pomocou <i>routing</i> funkcie . . . . .	25
Obrázok 14	Trasa s nedostatočne hustým počtom bodov . . . . .	26
Obrázok 15	Znázornenie interpolácie bodov . . . . .	27
Obrázok 16	Optimalizácia pripnutia trasy k cestnej sieti pomocou interpolácie bodov . . . . .	27

# Zoznam skratiek

<b>AI</b>	Artificial Intelligence
<b>API</b>	Application programming interface
<b>GPS</b>	Global Positioning System
<b>IDE</b>	Integrated Development Environment
<b>KB</b>	Kilobyte
<b>MVP</b>	Minimum viable product
<b>SQL</b>	Structured Query Language

# Zoznam algoritmov

# Zoznam výpisov

1	Dáta požiadavky na <i>Valhalla</i> kontajner . . . . .	29
2	Odpoveď z <i>Valhalla</i> kontajnera . . . . .	29
3	<i>retDict</i> premenná . . . . .	31

# Úvod

V dnešnej dobe je veľmi dôležité sledovanie polohy používateľov pomocou GPS. Či už sa jedná o GPS údaje zozbierané inteligentným mobilným zariadením, inteligentnými hodinkami, automobilom alebo iným zariadením. Tieto zozbierané údaje sú užitočné na niekoľko rôznych súvislostí, napríklad navigácia a mapovanie, aktivita a trasy alebo analytika a dátový výskum.

Dáta získané GPS senzormi však nie sú vždy dostatočne presné. Väčšina zariadení, ktoré snímajú GPS údaje sú nepresné, pretože nie sú určené primárne na toto využitie. Trasy zobrazené takýmito nepresnými senzormi vyzerajú neprehľadne a kvôli svojej neprehľadnosti neodovzdajú takmer žiadnu informáciu.

V našej práci sa budeme zaoberať tým, ako tieto nepresné trasy upraviť a zobraziť tak, aby používateľ dokázal zistiť pohybové vzory pohľadom. Trasy bude možné zobraziť na interaktívnej mape, ktorá sa bude dať priblížiť a oddialiť. Trasy bude možné zobraziť po nahratí do aplikácie. Trasy bude možné zobraziť všetky naraz alebo jednotlivo.

Práca bude obsahovať oboznámenie sa s problematikou vývoja webových aplikácií, výber vhodného vyvojového prostredie a vhodného programovacieho jazyka, v ktorom budeme webovú aplikáciu programovať. Ďalej obsiahne nejaké už existujúce aplikácie, v ktorých analyzujeme chyby a nedostatky, ktoré sa budeme snažiť v našom riešení odstrániť. Urobíme návrh riešenia a implementujeme aplikáciu. Vo finálnej forme aplikáciu otestujeme, či funguje správne a podľa očakávaní.

Cieľom práce je vytvoriť webovú aplikáciu, do ktorej používateľ nahrá GPS trajektóriu, ktoré budú spracované algoritmom a pripnuté k ceste, aby sa dali zobraziť prehľadne. Webová aplikácia následne zobrazí mapu s jednotlivými trasami. Zobrazené trasy budú informovať o spôsobe dopravy, názve nahraného súboru a času nahrania.

# 1 Problematika vývoja webových aplikácií

## 1.1 Webové aplikácie a ich vývoj

Webové aplikácie a ich vývoj je veľmi dôležitý v oblasti informačných technológií. Umožňuje vývojárom tvoriť a distribuovať softvér cez internet. Výhodou Webových aplikácií je, že poskytujú interaktívne a dynamické prostredie pre používateľov prostredníctvom internetových prehliadačov. Pri vývoji webových aplikácií je možné sa stretnúť s mnoha technológiami a programovacími jazykmi[1]:

- **HTML/CSS** Patria medzi základné jazyky, pomocou ktorých sa tvorí obsah webových stránok.
- **Javascript** Skriptovací jazyk, ktorý zaručuje dynamickosť webovej stránky, používateľ môže mať pomocou tohto jazyka interakciu so stránkou.
- **Backend jazyk** Jazyk, ktorý je použitý na serverovú časť webovej aplikácie, ako napríklad prihlásenie, prácu s databázou a inej logiky na pozadí, ktorú používateľ nevidí. Patria sem napríklad: Javascript, Python, PHP, Java a ďalšie.
- **Databázy** V mnoho aplikáciach treba dátá ukladať a tieto dátá sa ukladajú v databázach. Uložené dátá je možné prehľadávať alebo inak s nimi manipulovať. Medzi rôzne databázy patria napríklad MySQL, PostgreSQL, MongoDB a ďalšie.
- **Knižnice a frameworky** Používajú sa pre zjednodušenie vývoja a zlepšenie efektivity. Patria sem napríklad Angular, React, Vue.js pre frontend alebo Django, Laravel, Express, Flask pre backend.

## 1.2 Vývojový proces v 8 krokoch:

1. **Prieskum trhu:** V prvom rade treba zistíť, aký problém používateľ má. Treba naštudovať existujúce riešenia problému. Pri existujúcich riešeniach treba zvážiť výhody a nevýhody riešení. Vo vlastnej webovej aplikácii sa budeme snažiť nevýhody odstrániť.
2. **Definovanie funkcionality:** Aplikácia musí mať určené, aké funkcionality bude ponúkať. Tieto funkcionality majú riešiť problém používateľa. Chceme dosiahnuť, aby naša aplikácia riešila všetky problémy používateľa a aby nemala zbytočnú funkcionality, ktorú používateľ nebude používať a bude nevyužitá. Touto zbytočnou funkcionalityou sa proces vývoja zbytočne predĺži.

3. **MVP:** Vytvorenie základnej verzie aplikácie, kde sa nedáva špeciálny dôraz na dizajn. Je to návrh ako by mala aplikácia fungovať s implementáciou hlavných funkcií.
4. **Databáza:** Vytvorenie a pripojenie aplikácie na databázu, kde sa budú dátá bezpečne ukladať pre neskôršie načítanie a používanie v aplikácii.
5. **Frontend:** V tomto kroku vytvoríme dizajn aplikácie. S týmto dizajnom budú používatelia pracovať, preto by mal byť prehľadný, pekný a zaujímavý.
6. **Backend:** Vytvorenie serverovej časti. Patrí sem komunikácia s databázou, výpočty na pozadí, ktoré používateľ nevidí.
7. **Testovanie:** Testovanie aplikácie, či funguje správne. Odskúšame funkciu aplikácie a odhalíme prípadné chyby a zistíme, či sa aplikácia správa podľa očakávaní.
8. **Nasadenie:** Nahranie aplikácie na server, aby ju mohol používať používateľ odkialkoľvek.

### 1.3 Výber technológií

#### Javascript[2]

Javascript je jedným z najpopulárnejších a najuniverzálnejších programovacích jazykov pre vytváranie webových aplikácií. Umožňuje dynamické a interaktívne funkcie, ktoré zlepšujú používateľskú skúsenosť s aplikáciou a funkčnosť webových stránok. Môže bežať na strane klienta aj na strane servera. Je to interpretovaný jazyk. Má bohatý ekosystém knižníčok a frameworkov, ktoré poskytujú rôzne nástroje a abstrakcie pre vývoj webových aplikácií. Jedným z týchto frameworkov je aj Express, ktorý sme použili na vytvorenie serverovej časti. *Express* bližšie opíšeme v sekcii 3.5.

#### Visual Studio Code

Spomedzi viacerých vývojových prostredí a textových editorov, v ktorých je možné vytvárať webové aplikácie sme si vybrali *Visual Studio Code*. *Visual Studio Code* je textový editor, ktorý podporuje množstvo rozšírení a programovacích jazykov. Prináša mnoho výhod, ktoré sú opísané nižšie v sekcii 3.6.

#### MySQL

Aplikačné údaje, napríklad údaje o registrovaných používateľoch, informáciu o prihlásení a podobne ukladáme do aplikačných databáz. Tieto databázy umožňujú štrukturalizáciu

dát do tabuľiek, čo pomáha efektívnej správe informácií. *MySQL* je najpopulárnejšia open-source databáza[3]. Použili sme ju na uloženie informácií o používateľoch a na uloženie informácie pre správcu aplikácie v prípade zlého pripnutiu trasy k cestnej sieti. Viac o *MySQL* je popísané v sekcií 3.7.

### **Valhalla**

Existuje viaceré služieb, ktoré pripínajú trasy k cestnej sieti, napríklad MapBox, Google Maps alebo Valhalla. Valhallu sme si vybrali, pretože je open-source, dokáže pracovať s veľkým množstvom dát a nastavenia priprinania trasy k cestnej siete sú dostatočne nastaviteľné. To znamená, že vieme meniť parametre ako napríklad presnosť GPS zariadenia, dĺžka polomeru okruhu, v ktorom sa od bodu hľadajú kandidáti na pripnutie alebo penalizácia odbočovania na cestnej sieti. Všetky tieto parametre pomáhajú k lepšiemu pripnutiu trasy k cestnej sieti. Viac o *Valhalla* si v sekcií 3.2.

### **Leaflet**

Aby sme mohli na mape zobraziť trasy, potrebujeme najprv v našej aplikácii zobraziť mapu. Na toto sme zvolili Leaflet. Leaflet je open-source knižnica napísaná v *Javascripte*. Ponúka vývojárom silnú a flexibilnú platformu na vytváranie interaktívnych máp pre webové aplikácie. Je jednoduchá na použitie a pre našu webovú aplikáciu ideálna. Viac o *Leaflet* v sekcií 3.8.

## 2 Podobné aplikácie

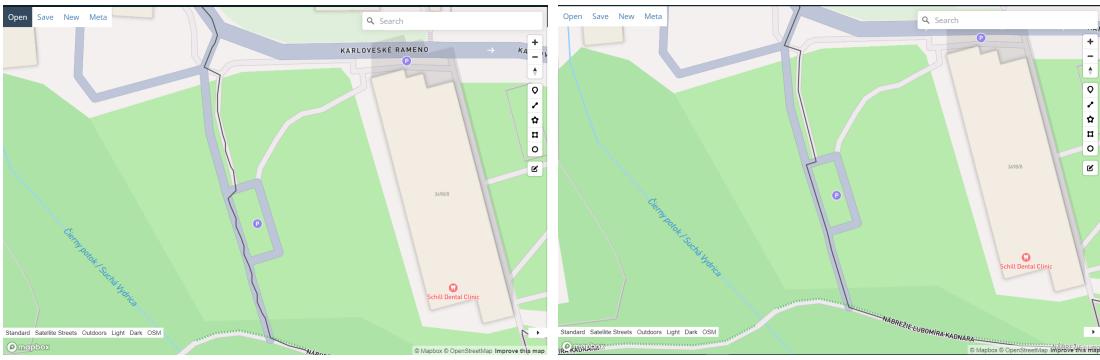
### 2.1 Mapbox

Mapbox je platforma údajov o polohe pre mobilné a webové aplikácie[4]. Poskytuje mnoho produktov: *Mapbox GL JS* knižnicu, pomocou ktorej je možné vytvoriť interaktívnu, prispôsobenú a efektívnu mapu vo webovej aplikácii[5], *Navigation SDK for mobiles*, riešenie pre vytváranie navigácie v mobilných telefónoch dostupné pre *Android* aj *iOS*[6], *MapGPT*, prvého AI asistenta, s ktorým je možné mať konverzácie ohľadom ciest, navigačných inštrukcií alebo atrakcií[7]. Výhody Mapboxu:

- Ponúka 50 tisíc načítaní mapy vo webovej aplikácii.
- Poskytuje map-matching, ktorý pripne vstupné GPS údaje trasy k cestnej sieti pre zaručenie prehľadného zobrazenie trás, to znamená, že trasy nebudú zobrazené mimo cesty ale budú ležať na cestnej sieti.
- Prispôsobenie štýlov mapy, markerov alebo vyskakovacích okien.

Nevýhody

- Neposkytuje vlastnú webovú aplikáciu, na ktorú je možné nahrať trasu, ktorú chceme zobraziť. Trasa sa dá zobraziť až po použití Mapbox GL JS a implementovaní vlastnej webovej aplikácie.
- Vstupné dátá treba upraviť na požadovaný formát pred odoslaním do Mapbox API.
- Do jednej požiadavky na map-match je možné vložiť len 100 bodov, preto je potrebné pre dlhšie trasy (tisíc bodov a viac) urobiť niekoľko požiadaviek. Pri aplikácií, ktorú používa veľa používateľov (rádovo tisíc) môže rýchlo dôjsť k vyčerpaniu kvóty pre map-match (100 tisíc mesačne) zadarmo. Taktiež rozdelenie trás do viacerých požiadaviek spomaluje rýchlosť map-matchingu.
- Trasa je ku cestnej sieti pripnutá nepresne, viď obrázok 1.
- Je potrebné použiť online API, čo môže spomalovať map-matching.



(a) Map-match od Mapbox

(b) Map-match od Valhally

Obr. 1: Porovnanie Mapbox a Valhalla map-matching

## 2.2 GPS Visualizer

GPS Visualizer je online nástroj, ktorý vytvára mapy a profily z geografických údajov. Lahko sa používa, je prehľadný a rýchly, to znamená, že pri prvom pohľade na nástroj používateľ vie, ako nástroj používať. Vstup môže byť vo forme údajov GPS (trasy a body na ceste), jazdných trás, ulíc alebo jednoduchých súradníc. Používa sa na zobrazenie trás, plánovanie trasy alebo na rýchlu vizualizáciu geografických údajov (na vedecké pozorovania, zobrazenie nehnuteľností, geograficky označených fotografií a podobne). Na webe je od októbra 2002[8]. Výhody:

- Rýchly, prehľadný a ľahký na použitie.
- Dokáže načítať dátá z rôznych zdrojov (GPX, URL, FIT, CSV, TRK a ďalšie).
- Mapu so zobrazenou trasou je možné stiahnuť vo formáte HTML stránky na neskôršie zobrazenie.

Nevýhody:

- Neponúka map-matching, iba zobrazuje trasy.
- Pre zobrazenie trasy je nutné zakaždým nahrať súbor s GPS údajmi.
- Nutnosť pripojenia na internet kvôli načítaniu knižníc.

Prostredie GPS visualizer je zobrazené na obrázkoch nižšie 2.



Obr. 2: GPS Visualizer prostredie

## 2.3 Routiak

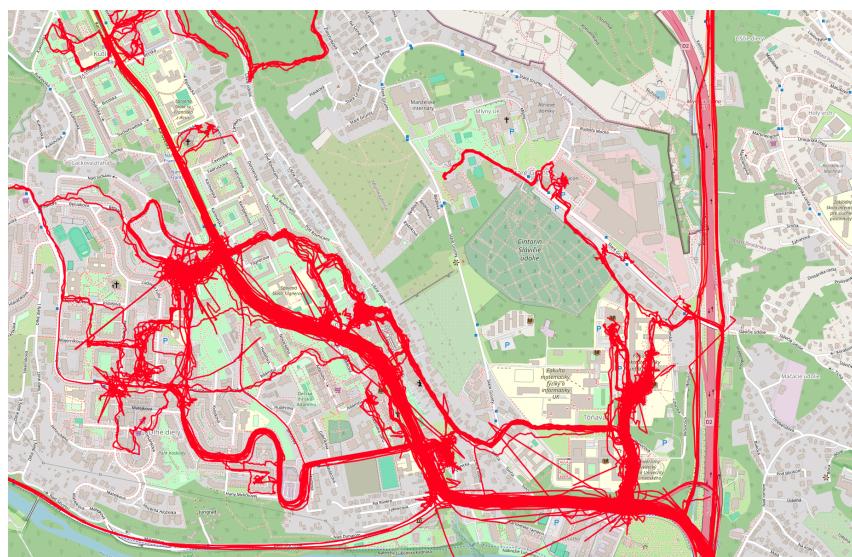
Routiak je webová aplikácia, pomocou ktorej si môže používateľ vizualizovať trasy na mape prehľadne. To znamená, že aj pri veľkom počte trás (10 a viac) prechádzajúcich tým istým miestom, môže používateľ nastaviť zobrazenie tak, aby bolo prehľadné a na mape nevznikali tzv. "špagety" (obr.3). Vďaka správnemu nastaveniu zobrazenia je možné vidieť trasy a cestnú sieť prehľadne. Používateľ sa môže v aplikácii registrovať a prihlásiť a teda je možné nahrané trasy zobraziť neskôr bez nutnosti znova trasy do aplikácie nahrať. Trasy sa do aplikácie nahrávajú v ZIP súbore, čo umožňuje nahrať viac trás súčasne. Tieto trasy je možné zobraziť na mape jednotlivo, alebo všetky naraz obsiahnuté v ZIP súbore. Aplikácia ponúka map-match funkcionality, čo znamená, že trasy budú pripnuté na cestnú sieť a tým sa zlepší viditeľnosť mapy a trás na mape, obrázok 4.

Výhody:

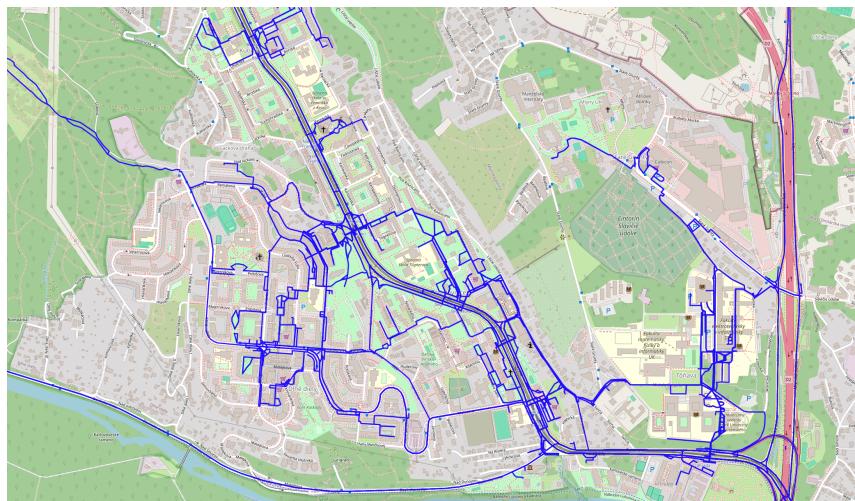
- map-match, pripnutie trás na cestnú sieť pre zlepšenie viditeľnosti
- možnosť nahrať viac trás naraz
- zobrazenie vlastných trás po nahratí bez nutnosti nahrávať trasy pri každom zobrazení
- možnosť nahrať trasy vo viacerých formátoch (CSV,GEOJSON,GPX)
- neobmedzený počet zobrazení mapy s neobmedzeným počtom nahraných súborov (počet nahraných súborov je obmedzený veľkosťou vyhradeného miesta na serveri)
- do jednej požiadavky na map-match môže íst až 20 tisíc súradníc
- informácia pre používateľa v prípade, že niektorá z trás nemala požadovaný formát

Nevýhody:

- jedna trasa môže obsahovať maximálne 20 tisíc bodov
- map-match pri pešej chôdzii nie úplný (viac v kapitole TODO)
- priebeh map-matchu ZIP súboru nie je zobrazený, iba jeho dokončenie
- aj pri nahratí jednej trasy je nutné vložiť trasu do ZIP súboru so špeciálne určenou štruktúrou (TODO zobraziť odkaz na štruktúru)



Obr. 3: Zobrazenie väčšieho množstva trás na rovnakom mieste



Obr. 4: Zobrazenie väčšieho množstva trás upravených map-matchom na rovnakom mieste

# 3 Použité technológie

## 3.1 Docker

Docker je softvérová platforma, ktorá umožňuje rýchlo vytvárať, testovať a nasadzovať aplikácie. Docker balí softvér do štandardizovaných jednotiek nazývaných kontajnery, ktoré majú všetko, čo softvér potrebuje na spustenie vrátane knižníc, systémových nástrojov, kódu a runtime. Pomocou Dockeru môžete rýchlo nasadiť a škálovať aplikácie do akéhokoľvek prostredia a vediet, že váš kód sa spustí.[9]. Docker sa využíva z rôznych dôvodov[10]:

- **Spravovanie závislostí:** Docker umožňuje špecifikovať a spravovať všetky závislosti aplikácie v súvisiacom kontajneri, čo uľahčuje riadenie verzií a aktualizácií.
- **Rýchlosť nasadzovania:** Kontajnery sa rýchlo spúšťajú a zastavujú, čo zjednodušuje vývoj a nasadzovanie aplikácií.
- **Izolácia:** Kontajnery izolujú aplikáciu a jej závislosti od zvyšku systému, čo zabezpečuje konzistentné a predvídateľné spustenie aplikácie bez ohľadu na prostredie.
- **Portabilita:** Kontajnery sú prenosné medzi rôznymi prostrediami, čo znamená, že sú spustiteľné na rôznych serveroch, vývojových počítačoch a v cloude bez problémov.
- **Škálovateľnosť:** Docker kontajnery môžu byť ľahko škálovateľné pridávaním viacerých inštancií alebo zvyšovaním výkonu existujúcich inštancií.

## 3.2 Valhalla api

Valhalla je open source trasovací engine so sprievodnými knižnicami na použitie s dátami OpenStreetMap. Valhalla tiež obsahuje nástroje ako výpočet matice času a vzdialenosť, izochróny, vzorkovanie nadmorskej výšky, porovnanie máp a optimalizácia trás (Travelling Salesman)[11].

### 3.2.1 Odin (generovanie trasy, rozpoznávanie manévrov)

Odin slúži ako nástroj inštrukcii na anotáciu cesty ako vstup zo smerovacieho enginu na použitie v navigácii. V súlade so severskou mytologickou térou bolo zvolené meno Odin, pretože sa o ňom často hovorilo, že je veľmi mûdry. Keďže knižnica sa zaobrá väčšinou poskytovaním rozumného navádzania po ceste, ktoré sa má použiť na navigáciu, zdalo sa to ako vhodné meno. Odin obsahuje súbor rôznych dátových štruktúr a algoritmov, ktoré sa zaobrajú vecami ako: generovanie manévrov, porovnanie názvov ulíc a generovanie rozprávania. [12]

### **3.2.2 Thor (trasovacie algoritmy)**

Thor slúži ako trasovací mechanizmus podporovaný trasovacími dátami s otvoreným zdrojom. Thor je spoločníkom Sif, na ktorý sa veľmi spolieha pri určovaní vhodného prechodu grafom. Výsledná cesta môže byť použitá ako vstup pre vytvorenie manévr. Meno Thor bolo zvolené ako skratka pre: *Tiled Hierarchical Open Routing* a bola základnou myšlienkou, okolo ktorej sa sformovala organizácia Valhalla a jej téma severskej mytológie.

Knižnica Thor je v podstate súbor rôznych dátových štruktúr a algoritmov, ktoré sa zaoberajú vecami ako: prechod grafom A\*, kalkulácia hrán, kalkulácia vrcholov a konštrukcia cesty. Zahŕňa tiež metódy na výpočet matíc časových vzdialenosí, optimalizované trasovanie a izochróny.

### **3.2.3 Sif (cena pre hrany a prechody)**

Sif poskytuje dynamickú, rozšíriteľnú kalkuláciu pre hrany a prechody medzi hranami (náklady na odbodenie). Jeho primárne využitie je v trasovacom engine pri vytváraní najlepšej cesty medzi miestami. V súlade so severskou mytologickou térou bolo zvolené meno Sif, pretože Sif je spoločníkom Thora.

Sif je v podstate súbor rôznych dátových štruktúr a algoritmov, ktoré sa zaoberajú vecami ako: korelácia vstupného miesta so základným grafom, čiastočná vzdialenosť pozdĺž hrany a filtrovanie hrán, ktoré by sa nemali brať do úvahy pri korelácii.

### **3.2.4 Loki (priradenie lokácie grafovým vrcholom)**

Loki sa používa na priradenie informácií o polohe k základnému objektu grafu na použitie pri vytváraní vstupu do trasovacieho enginu. V súlade so severskou mytologickou térou bolo meno Loki vybrané ako slovná hračka nájsť. Kedže Loki sa zaoberá väčšinou koreláciou nejakého vstupu (minimálne zemepisnej šírky, dĺžky) k objektu v rámci grafu, zdalo sa to ako vhodný názov!

Loki je v podstate súbor rôznych dátových štruktúr a algoritmov, ktoré sa zaoberajú vecami ako: korelácia vstupného miesta so základným grafom, čiastočná vzdialenosť pozdĺž okraja a filtrovanie hrán, ktoré by sa nemali brati do úvahy pri korelácii.

### **3.2.5 Mjolnir (trasovací graf a generovanie dlaždíc)**

Mjolnir je v súbor aplikácií, dátových štruktúr a algoritmov, ktoré sa zaoberajú vecami ako: spracovanie dátových extraktov OSM, rezanie smerovateľných „grafových“ dlaždíc, generovanie hierarchií dlaždíc a testovanie na nedostatky údajov. Tieto dlaždicové smerovacie údaje sa používajú pri smerovaní a vyhľadávaní v rámci organizácie Valhalla. V súlade so severskou mytologickou térou bol zvolený názov Mjölnir, pretože predstavuje zbraň hromadného ničenia. Zdalo sa to vhodné, pretože hlavná aplikácia sa zaoberá väčšinou

rozbíjaním údajov OSM veľkosti planéty do malých fragmentov smerovateľných dlaždíc.

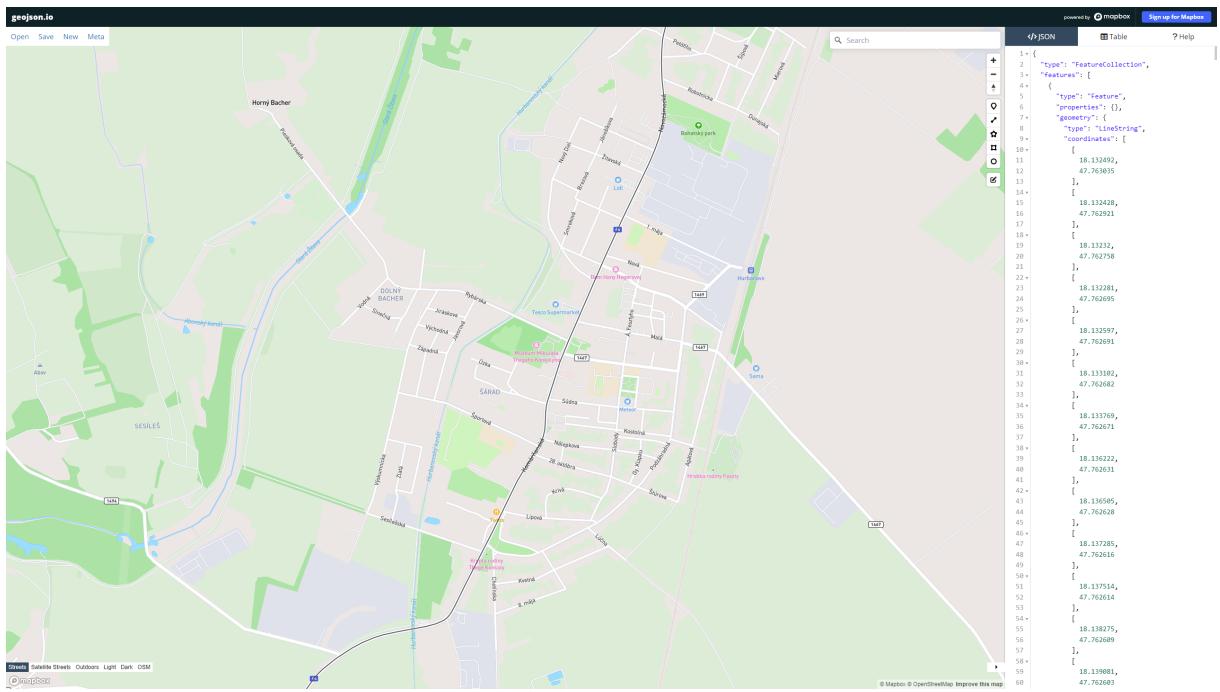
### 3.2.6 Meili (map matching)

Meili, poskytuje sadu algoritmov a dátových štruktúr na porovnávanie máp. Priraduje sekvenciu miest (zvyčajne nepresných, napr. trajektória GPS) k základnej cestnej sieti. V súlade so severskou mytologickou tematikou bolo zvolené meno Meili, Thorov brat. Keďže porovnávanie máp úzko súvisí so smerovaním a keďže Thor je trasovacia knižnica Valhally, Meili sa zdalo najvhodnejšie.

## 3.3 Geojson.io

GeoJSON.io je webový nástroj, ktorý umožňuje používateľom vytvárať, upravovať a vizualizovať údaje GeoJSON. GeoJSON je formát na kódovanie geografických dátových štruktúr, ako sú body, čiary a polygóny, pomocou JavaScript Object Notation (JSON). Bežne sa používa na reprezentáciu priestorových informácií a je podporovaný mnohými nástrojmi mapovania a GIS (Geografický informačný systém). Hlavnými výhodami sú:

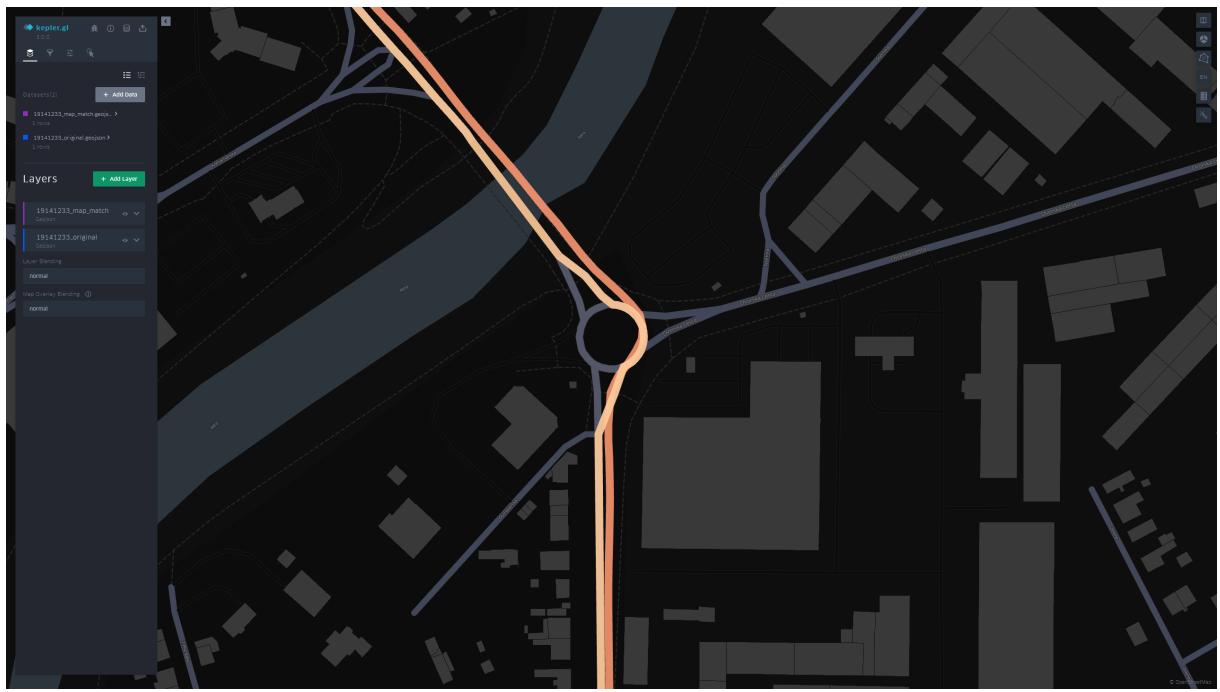
- **Interaktívne rozhranie:** Nástroj poskytuje interaktívnu mapu, na ktorej si používateľ dokáže nakresliť a upraviť GeoJSON.
- **Import/Export dát:** Nástroj umožňuje jednoducho nahrať a zobrazit GeoJSON, alebo uložiť GeoJSON vytvorený v nástroji.
- **Dizajn:** Jednoduchý a intuitívny dizajn pre rýchle použitie bez inštalácie.



Obr. 5: Prostredie Geojson.io

### 3.4 Kepler.gl

Kepler.gl je open source nástroj na geopriestorovú analýzu vyvinutý spoločnosťou Uber. Ide o webovú aplikáciu, ktorá používateľom umožňuje vizualizovať a skúmať rozsiahle súbory geolokačných údajov. Kepler.gl je údajovo agnostický a dokáže vykresliť milióny bodov reprezentujúcich tisíce ciest, čo z neho robí výkonný nástroj na priestorovú analýzu a vizualizáciu. Je postavený na platforme deck.gl WebGL na vizualizáciu údajov a možno ho okamžite použiť z webového prehliadača bez toho, aby ste museli čokoľvek inštalovať. Kepler.gl je v Uberi široko používaný na podporu pokročilých geopriestorových analýz inžiniermi, analytikmi a dátovými vedcami[13][14][15].



Obr. 6: Prostredie Kepler.gl

### 3.5 Express (Node.js)

Express je backend aplikačný framework na vytváranie serverovej časti webových aplikácií. Je vydaný ako bezplatný softvér s licenciou MIT. Dá sa nazvať štandardným serverovým frameworkom pre Node.js[16]. Jeho prvá verzia, aspoň podľa histórie Githubu, bola dňa 22. Mája 2010. Prvá vydaná verzia bola 0.1.0 v Novembri 2010[17]. Express je napísaný v Javascripte a beží na Node.js platforme.

Výhody Express[18]:

- **Jednoduchý na použitie:** Express je účinný a užívateľsky prívetivá platforma. Má intuitívny API a jednoduchú štruktúru. Umožňuje zostaviť aplikáciu len pomocou párov riadkov kódu, ktoré majú priamu a minimalistickú syntax.
- **Veľká komunita:** Mnoho vývojárov používa tento framework, takže je veľmi populárny a existuje množstvo návodov, pomôcok a fórum, kde sa dá obrátiť pri probléme pri vývoji. Taktiež má množstvo rozšírení, middleware a ďalších nástrojov, ktoré môžu byť integrované do aplikácií.
- **Správanie HTTP požiadaviek:** Vďaka jednoduchému a flexibilnému spôsobu definovanie ciest pre rôzne HTTP požiadavky môžeme ľahko určiť ako bude naša aplikácia reagovať a pracovať s dátami podľa našich potrieb.

- **Škálovateľnosť:** Express je ľahko škálovateľný, takže pri vhodnom návrhu aplikácie môžeme efektívne spracovať aj veľké množstvo požiadaviek.

Nevýhody Express[18]:

- **Bezpečnosť:** Bolo ohlásených mnoho bezpečnostných problémov, ako napríklad vzdialené spustenie kódu pomocou *ejs* modulu alebo častný *denial of service* pri čerpstvom module.
- **Aktualizácie:** Kedže sa tento framework stále vyvíja a zlepšuje, môže nastať, že niektoré moduly už nebudú podporované alebo sa budú používať inak ako sa očakáva. Preto je potrebné kontrolovať aktualizácie a zmeny a uistit sa, že tieto aktualizácie neprinesú do našej aplikácie nové chyby.

Použitie Express v populárnych aplikáciách: Uber, Netflix, PayPal a LinkedIn [18].

### 3.6 Visual Studio Code

Visual Studio Code je jednoduchý, ale výkonný editor zdrojového kódu, ktorý je stiahnuťelný pre Windows, MacOS aj Linux. Dodáva sa so vstavanou podporou pre JavaScript, TypeScript a Node.js [19]. Má bohatý ekosystém rozšírení pre ďalšie jazyky. Taktiež obsahuje množstvo pluginov a rozšírení. Ponúka množstvo vývojárskych nástrojov ako integrácia gitu, debugger alebo vstavaný príkazový terminál. Bol vydaný 29. Apríla 2015 [20]. Jeho hlavnými výhodami sú [21]:

- Je zadarmo.
- Dostupný pre Windows, MacOS aj Linux.
- Je rozšíriteľný a je možné nainštalovať rozšírenia pre takmer všetky populárne programovacie jazyky. Nie je potrebné stahovať zvlášt IDE pre každý programovací jazyk, s ktorým pracujete.
- Pracuje rýchlejšie aj veľkým množstvom pluginov ako ostatné IDE.
- Je bohatý na funkcie a s vhodnými nainštalovanými rozšíreniami je porovnatelný s plnohodnotnými IDE aj napriek tomu, že ide stále len o textový editor.

```

File Edit Selection View Go Run Terminal Help
DPI
EXPLORER
  -> DFI
    -> webapp
      -> public
        -> db.sql
        -> index.js
        -> map_match.py
        -> map.js
        -> package.json
        -> upload.js
        -> .gitignore
        -> Dockerfile
        -> README.md
        -> OUTLINE
        -> TIMELINE
index.js
1  <!-- index.js -->
2  webapp > index.js > (app.post('/register') callback
3  connectToDatabase();
4
5  // Route to handle user registration
6  app.post('/register', async (req, res) => {
7    const { username, password } = req.body;
8    try {
9      const hashedPassword = bcrypt.hashSync(password, 10);
10     const sql = 'INSERT INTO users (username, password) VALUES (?, ?)';
11     connection.query(sql, [username, hashedPassword], (err, result) => {
12       if (err)
13         console.error(`Error registering user: ${err.message}`);
14       res.status(200).send(err.message);
15       return;
16     });
17     res.status(200).send("SUCCESS");
18   }) catch (err) {
19     console.error(`Error registering user: ${err.message}`);
20     res.status(400).send(err.message);
21   }
22 }
23
24 app.post('/logout', (req, res) => {
25   req.session.username = null;
26   res.redirect('/login');
27 });
28
29 // Route to handle login
30 app.post('/login', async (req, res) => {
31   const { username, password } = req.body;
32   const sql = 'SELECT * FROM users WHERE username = ?';
33   connection.query(sql, [username], async (err, results) => {
34     if (err)
35       console.error(`Error fetching user: ${err.message}`);
36     res.status(500).send(err.message);
37     return;
38   });
39   if (results.length === 0) {
40     res.status(401).send('Invalid username or password');
41     return;
42   }
43   const user = results[0];
44   if (await bcrypt.compareSync(password, user.password)) {
45     req.session.username = user.username;
46     res.status(200).send('SUCCESS');
47   } else {
48     res.status(401).send('Invalid username or password');
49   }
50 });

```

Obr. 7: Prostredie Visual Studio Code

### 3.7 MySQL

Všetky softvérové aplikácie vyžadujú ako primárny zdroj údajov databázy. Databáza ukladá údaje, keď niekto vykoná online vyhľadávanie, prihlási sa do účtu alebo dokončí transakciu, aby bolo neskôr možné tieto údaje získať. Namiesto ukladania všetkých údajov do jedného veľkého súboru, uchováva relačná databáza údaje usporiadane do tabuľiek. Skutočné súbory, ktoré tvoria štruktúru databázy, sú usporiadane tak, aby maximalizovali rýchlosť. Je možné nastaviť vzťahy medzi jednotlivými údajmi 1:1, 1:N. Dátové polia môžu byť jedinečné, povinné alebo voliteľné. Databáza presadzuje tieto pravidlá, čo zaistuje, že v prípade dobre navrhnutej databázy vaša aplikácia nikdy nenarazí na nekonzistentné, duplicitné, ojedinelé, zastarané alebo chýbajúce údaje. Časť SQL v *MySQL* známená Structured Query Language. SQL je najbežnejší štandardizovaný jazyk používaný na prístup k databázam. Výhody *MySQL*[3]:

- Databáza je jednoduchá na inštaláciu, to znamená, že je vývojár nainštaluje do pár minút.
- Je ľahko škálovateľná. Zvládne operovať nad veľkým množstvom dát, rovnako ako obslúžiť veľké množstvo pripojení. Je preto vhodná aj pre veľké aplikácie ako *Facebook*.
- Poskytuje mnoho bezpečnostných prkov, napríklad autentifikáciu alebo nastavenie práv pre rôznych používateľov. Vďaka tomuto sú dátá v bezpečí pred neautorizovanými.

nými používateľmi.

### 3.8 Leaflet

*Leaflet* je poprená open-source knižnica napísaná v *Javascripte* pre interaktívne mapy či už pre mobilné telefóny alebo počítače. Spolu má menej ako 50KB a poskytuje všetky funkcie mapy, ktoré väčšina vývojávor potrebuje[22]. Je jednoduchá, prehľadná a dajú sa pomocou nej jednoducho vykreslovať trasy na mapu.

Výhody[23]:

- Jednoduchosť - *Leaflet* je dizajnovaný pre jednoduché použitie. Je možné vytvoriť mapu jednoduchým skopírovaním kódu z *QuickStart* tutoriálu [24].
- Dokumentácia - Dokumentácia pre *Leaflet* je dobre štrukturovaná s mnohými príkladmi a návodmi.
- Komunita - Kedže *Leaflet* je najpopulárnejšou *Javascriptovou* knižnicou pre prácu s mapami, má obrovskú komunitu používateľov. Chýbajúce príklady zložitejšieho využitia sú doplnené mnohými príkladmi používateľov z komunity. "leaflet stackoverflow" na *Google* vyhľadávaní vráti takmer 400 tisíc výsledkov.
- Flexibilita - V ponuke sú všetky funkcie, ktoré väčšina vývojárov pri práci s mapami potrebuje. V prípade potreby je možné funkcionality doplniť pluginmi.

# 4 Návrh riešenia

## 4.1 Špecifikácia požiadaviek

V tejto sekcií sa oboznámime s požiadavkami, ktoré má naša aplikácia splňať. Rozdelíme ich na funkcionálne a nefunkcionálne požiadavky. Taktiež si predstavíme diagram prípadov použitia a sekvenčné diagramy niektorých procesov.

### 4.1.1 Funkcionálne požiadavky

Funkcionálne požiadavky určujú, čo presne je možné v aplikácii robiť, alebo akú má funkciu.

#### Zoznam funkcionálnych požiadaviek

- **registrácia** - používateľ sa môže zaregistrovať pomocou používateľského mena a hesla
- **prihlásenie** - používateľ sa môže prihlásiť pomocou používateľského mena a hesla
- **odhlásenie** - používateľ sa môže z aplikácie odhlásiť
- **nahranie trás** - používateľ môže do aplikácie nahrať ZIP súbor obsahujúci trasy
- **zobrazenie nahraných trás** - používateľ môže zobraziť zoznam všetkých súborov, ktoré nahral a boli úspešne spracované aplikáciou
- **zobrazenie všetkých trás v nahranom súbore** - používateľ môže na mape zobraziť trasy, ktoré nahral v ZIP súbore
- **zobrazenie jednotlivých trás v nahranom súbore** - používateľ môže na mape zobraziť jednodlivo trasy, ktoré obsahuje ZIP súbor, ktorý nahral
- **pripnutie trás k cestnej sieti** - používateľ môže zobraziť trasy spracované aplikáciou, ktoré sú pripnuté k cestnej sieti. Zobraziť môže všetky trasy, ktoré obsahuje ZIP súbor, ktorý nahral, alebo jednotlivé.
- **zobrazenie chybových nahraných súborov** - v prípade, že nahraný súbor nemal správny formát alebo bola jeho štruktúra nevhodná, sa tento súbor uloží na neskôršie stiahnutie a používateľ je informovaný o chybe, ktorá nastala
- **znovu-spustenie chybných súborov** - v prípade, že sa nejaké trasy nachádzajú v priečinku chybových súborov, je možné na týchto trasách znova spustiť algoritmus pripnutia trás k cestnej sieti. Ide o špeciálny prípad použitia, keď si používateľ nahrá

na server súbory bez toho, aby ich dával do ZIP súboru. Týmto sa ušetrí čas pri komprimovaní trás do ZIP súboru a pri rozbalovaní ZIP súboru.

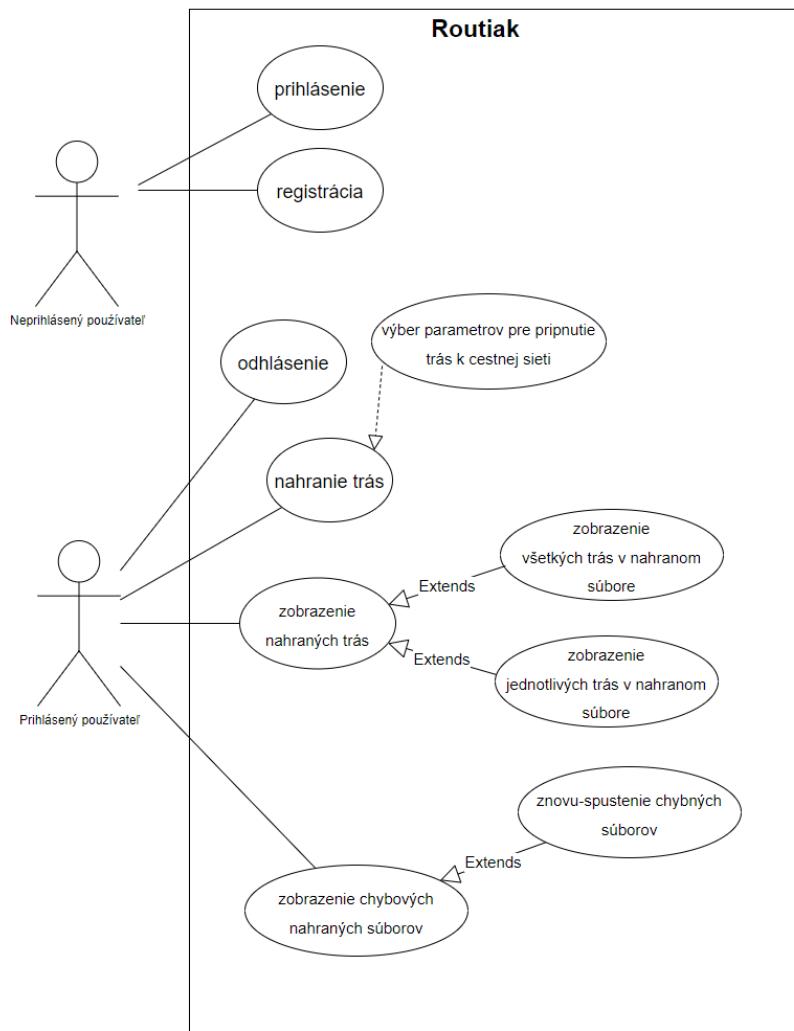
#### 4.1.2 Nefunkcionálne požiadavky

Nefunkcionálne požiadavky sa týkajú toho, ako má systém pracovať. Popisujú parametre systému a nie to, čo má systém robiť.

##### Zoznam nefunkcionálnych požiadaviek

- **jednoduchosť a prehľadnosť** - aplikácia má byť jednoduchá na použitie a po prečítaní návodu na použitie má byť jasné ako má byť použitá
- **prehľadný kód** - kód má byť písaný prehľadne, aby sa v budúcnosti dal jednoducho rozšíriť a aby bol udržateľný
- **jasná informácia o chybnej trase** - ak nastane chyba pri nahraní alebo spracovaní trás, používateľ má byť jasne informovaný, kde nastala chyba

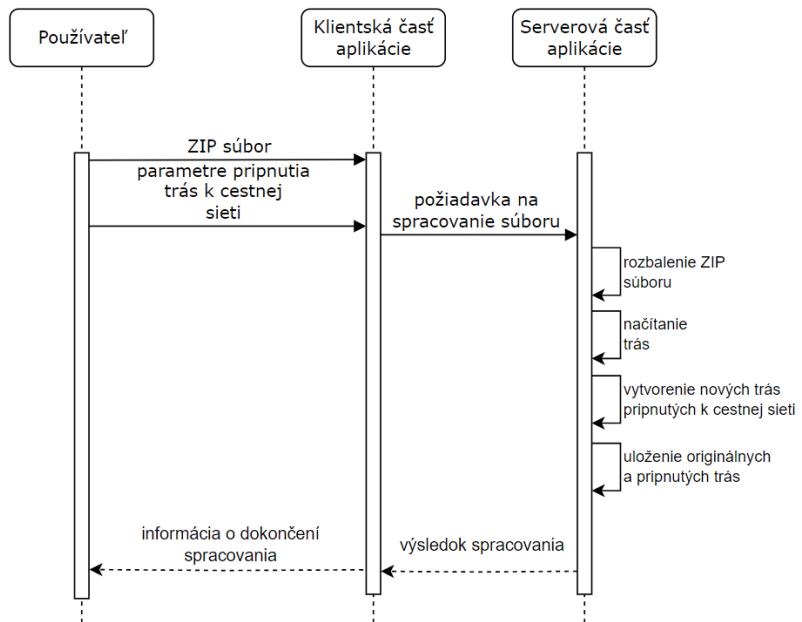
### 4.1.3 Diagramy



Obr. 8: Diagram prípadov použitia

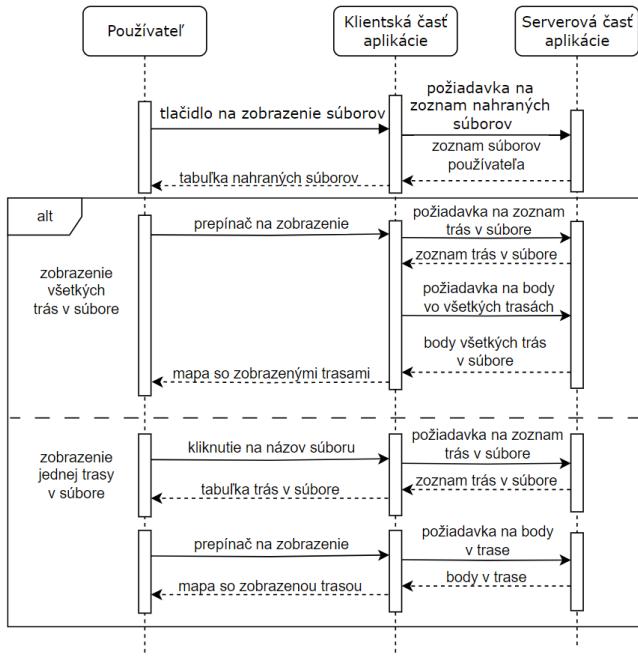
Používateľ sa bude musieť po otvorení webovej aplikácie prihlásiť alebo registrovať. Po prihlásení sa rola používateľa zmení na prihláseného. Ako prihlásený bude používateľ môcť nahrať trasy vo formáte ZIP. Pri nahrávaní súboru musí taktiež zvoliť parametre, ktoré budú použité systémom pri pripínaní trás k cestnej sieti. Prihlásený používateľ si bude môcť zobraziť trasy rozdelené podľa súborov, ktoré nahral. Trasy v týchto súboroch bude potom môcť zobraziť všetky naraz, alebo jednotlivo. Taktiež si bude môcť zobraziť chybové nahrané súbory v prípade, že nejaké chybové súbory nahral. Na týchto chybových súboroch bude potom môcť spustiť algoritmus pripínania trás k cestnej sieti znova. Nižšie je možné vidieť priebeh procesov nahrania súborov (obr.9), zobrazenia trás (obr.10) alebo zobrazenia chybových súborov (obr. 11) pomocou sekvenčných diagramov.

## Nahranie súborov

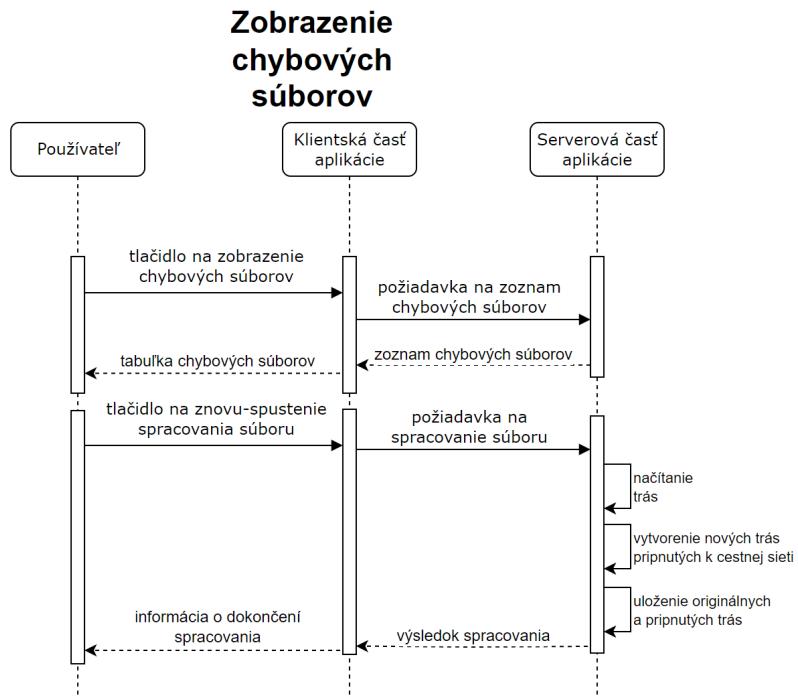


Obr. 9: Sekvenčný diagram procesu nahrania súborov na server.

## Zobrazenie trás



Obr. 10: Sekvenčný diagram procesu zobrazenia trás na mape.



Obr. 11: Sekvenčný diagram procesu zobrazenia chybových súborov

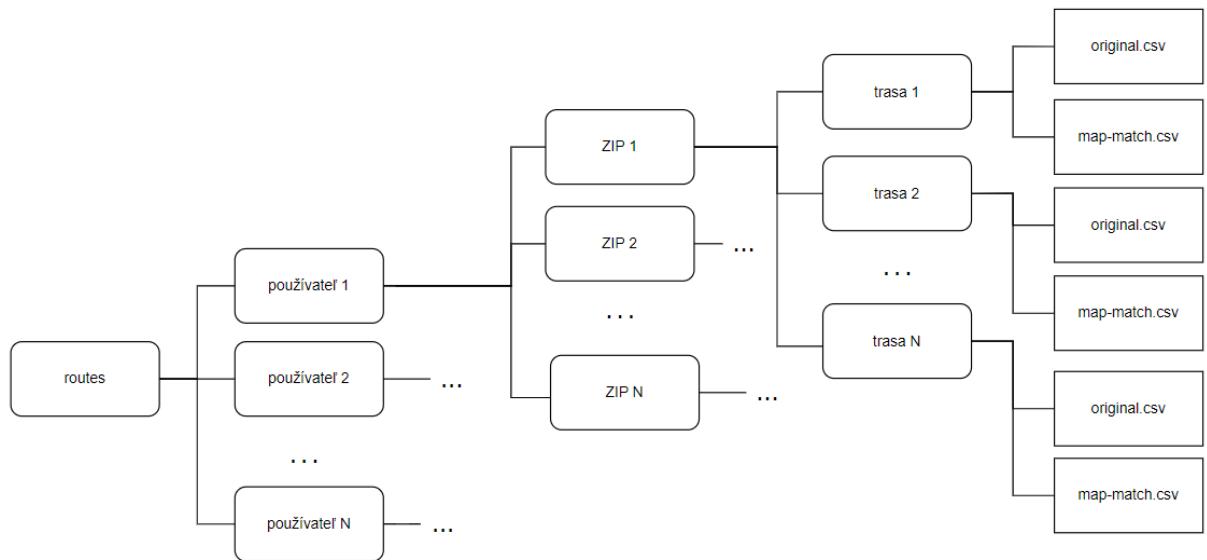
## 4.2 Ukladanie údajov

V aplikácii sú vytvorené priečinky *uploads* a *routes*. Do priečinka *uploads* sa budú ukladať všetky nahrané súbory. Po nahratí ZIP súboru na server bude pre používateľa vytvorený priečinok, ktorého názov bude zodpovedať prihlásovaciemu menu používateľa. V priečinku používateľa sa vytvorí nový priečinok *unzipped*, v ktorom sa vytvorí priečinok s názvom nahraného ZIP súboru. Do tohto priečinku sa rozbalí nahraný ZIP súbor. Cesta k tomuto priečinku sa uloží do premennej, s ktorou algoritmus neskôr bude pracovať. Po skončení algoritmu prebehne kontrola, ktorá určí či boli všetky trasy v ZIP súbore spracované správne. V prípade, že v ZIP súbore bude nejaká chybová trasa, ZIP súbor spolu s rozbalenými súbormi ostane na serveri pre neskôršie stiahnutie používateľom alebo prípadné znova-spustenie algoritmu. V opačnom prípade sa priečinok aj s nahraným ZIP súborom vymaže, aby sa ušetrilo miesto na serveri.

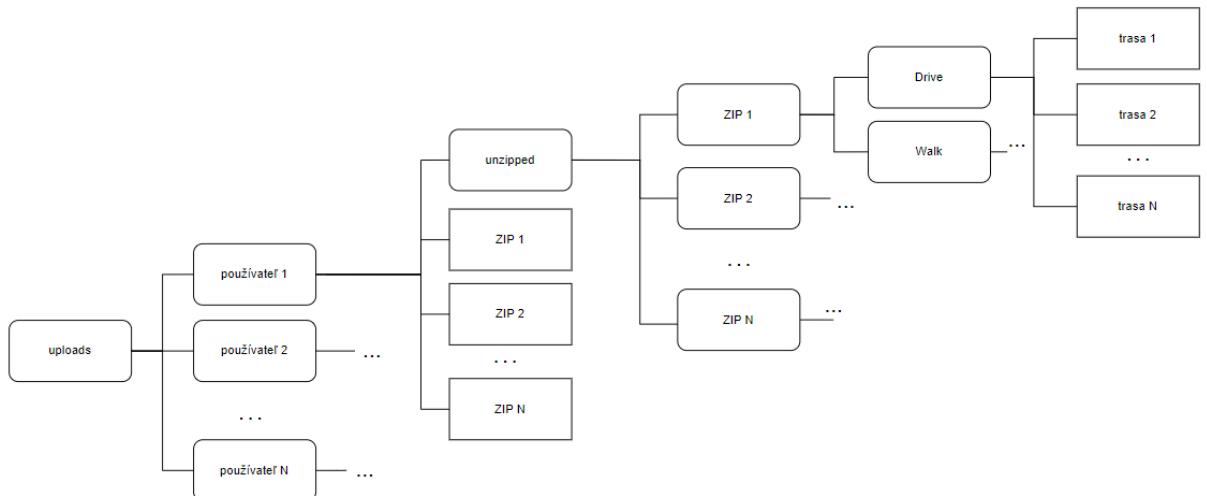
Štruktúra priečinka *routes* bude podobná priečinku *uploads*. Do priečinka *routes* sa po dokončení algoritmu vytvorí priečinok (ak ešte neexistuje), ktorého názov bude zodpovedať prihlásovaciemu menu používateľa. V priečinku používateľa bude vytvorený priečinok s názvom zodpovedajúcim názvu nahraného ZIP súboru. Do priečinku s názvom súboru budú vytvorené priečinky, ktorých názov bude zodpovedať názvom trás, ktoré boli nahrané v ZIP súbore a ktoré boli úspešne spracované algoritmom pripínania trás k cestnej sieti. V

každom takomto priečinku budú dva súbory s názvom *original.csv* a *map-match.csv*, ktoré budú obsahovať konkrétnie body trasy.

Štruktúry priečinkov sú vizualizované na obrázkoch . Priečinky sú zobrazené zaobleným obdĺžnikom a súbory sú zobrazené obdĺžnikom. Tri bodky "... " vertikálne medzi priečinkami a súbormi symbolizujú, že v priečinku môže byť 0 až N súborov. To znamená, že môže byť priečinok prázdny, alebo v ňom môže byť ľubovoľný počet súborov/priečinkov. Tri bodky "... " horizontálne za priečinkom symbolizujú, že graf pokračuje ale pre zjednodušenie obrázka bol graf skrátený.



(a) Štruktúra priečinku *routes*



(b) Štruktúra priečinku *uploads*

Obr. 12: Štruktúry priečinkov *uploads* a *routes*.

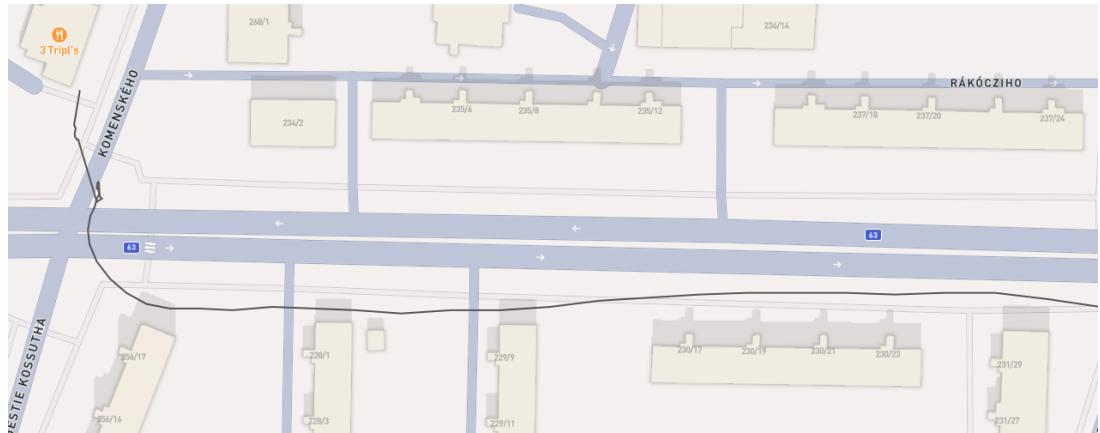
## 4.3 Pripínanie trás k cestnej sieti

### Základná myšlienka

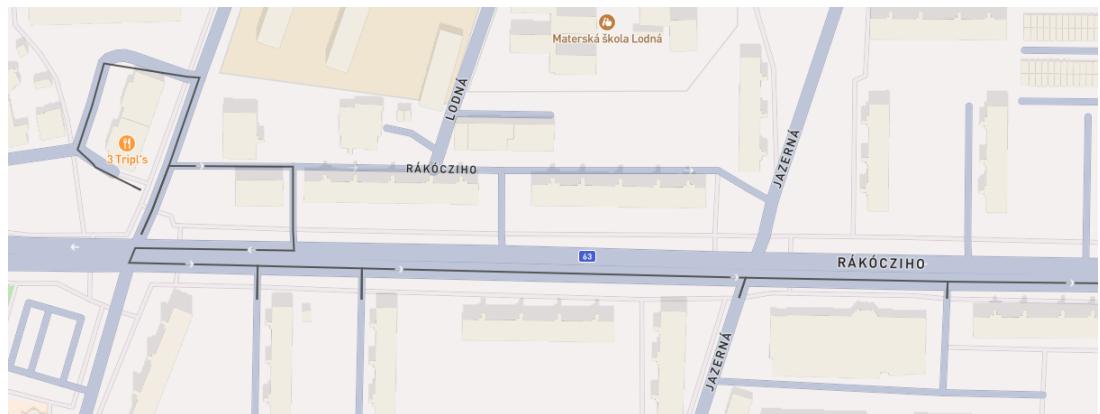
Po oboznámení sa s knižnicami, ktoré poskytujú funkciu pripínania trás k cestnej sieti sme dospeli k záveru, že trasy môžeme pripnúť dvoma spôsobmi. Pomocou funkcie *routing* alebo pomocou funkcie *map-match*.

Prostredníctvom funkcie *routing* by sme medzi každým bodom v trase našli optimálnu trasu, podobne ako v navigácii. Spojením týchto trás by sme dostali výslednú trasu pripnutú k cestnej sieti. Môže však nastať situácia, kedy *routing* funkcia kvôli nepresnosti GPS pripne bod k druhej ceste, viditeľné na obrázku 13b. Pre odstránenie takýchto situácií vytvoríme skript, ktorý odstráni body, ktoré sa v trase opakujú dva alebo viackrát. Takisto odstráni aj body medzi týmito dvoma opakujúcimi sa bodmi. Výslednok je možné vidieť na obrázku 13c. Po otestovaní zistili, že algoritmus odstraňujúci duplicitné body môže odstrániť aj dôležité body, ktoré by v trase mali ostať. Rozhodli sme sa v tomto riešení nepokračovať.

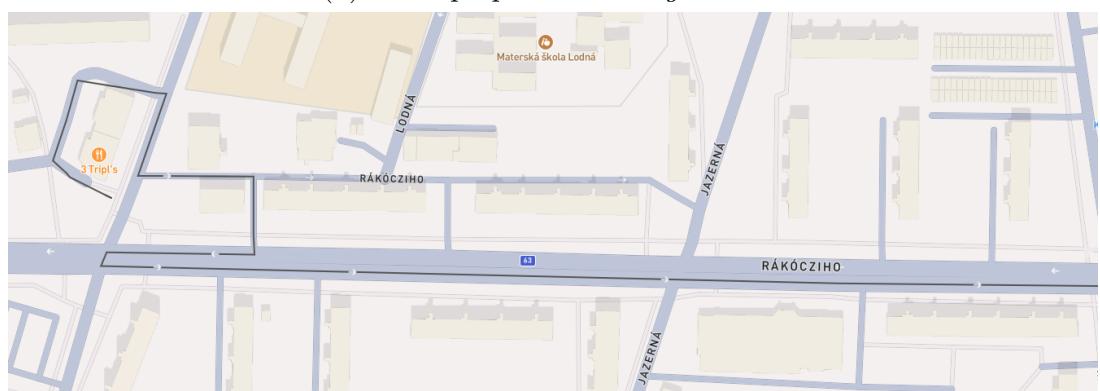
Druhou možnosťou je použiť funkciu *map-match*, ktorá body v trase posunie k cestnej sieti. V prípade, že body v trase nie sú dostatočne husté, môže byť výsledná trasa hranatá a jej tvar nemusí zodpovedať cestnej sieti. Toto je možné vidieť na obrázku 14b. Obrázok 14a znázorňuje trasu pred použitím *map-match* funkcie. Na obrázku 14b vidieť, že body trasy boli pripnúté k ceste, avšak nebolo ich dosť a výsledná trasa nezodpovedá cestnej sieti.



(a) Pôvodná trasa



(b) Trasa po použití *routing* funkcie



(c) Trasa po použití *routing* funkcie s odstránením duplicitných bodov

Obr. 13: Trasa priprnutá pomocou *routing* funkcie



(a) Trasa pred použitím *map-match* funkcie (b) Trasa po použití *map-match* funkcie

Obr. 14: Trasa s nedostatočne hustým počtom bodov

### Optimalizácia výsledkov

Pre dosiahnutie lepšieho pripnutia trasy k cestnej sieti body interpolujeme. To znamená, že medzi každú dvojicu po sebe idúcich bodov v trase vložíme ďalší bod do stredu medzi dva vybrané. Na obrázku 15 možno vidieť pôvodné body v modrom kruhu a pridané body označené červeným znakom. Týmto spôsobom dostaneme trasu obsahujúcu viac bodov, ktorá môže byť lepšie pripnutá k cestnej sieti.

Vyskúšali sme rôzne nastavenia interpolácie, napríklad opakovanie iterovanie pridávania bodov medzi pôvodné body, až pokial vzdialenosť medzi bodmi nebola menšia ako  $x$  metrov, pričom  $x$  bola premenná, ktorú sme menili. Po vyskúšaní rôznych nastavení interpolácie je viditeľné, že tvar trasy viac zodpovedá cestnej sieti. Táto optimalizácia však nie je dostatočná, preto vyskúšame inú. Rozdiel je pozorovateľný na obrázkoch 16a a 16b.



Obr. 15: Znázormenie interpolácie bodov



(a) Pred použitím interpolácie

(b) Po použití interpolácie

Obr. 16: Optimalizácia pripnutia trasy k cestnej sieti pomocou interpolácie bodov

# 5 Implementácia

## 5.1 Pripínanie trasy k cestnej sieti

Ukladanie trás spracovaných funkciou *map-match*, rovnako ako ukladanie pôvodných bodov trás prebieha pomocou skriptu, ktorý je napísaný v *Python*. Tento skript dostane na vstup slovník, ktorý obsahuje konfiguračné nastavenia skriptu, ako aj parametre pripínania trás k cestnej sieti. Medzi nastavenia skriptu patrí:

- unzipdir - cesta k priečinku, ktorý má byť spracovaný *map-match* algoritmom
- container - názov *Valhalla* kontajnera, ktorý poskytuje *map-match* funkcionalitu.
- user - meno používateľa, ktorý nahral súbor alebo požiadal o znova-spustenie algoritmu
- zipname - názov nahraného ZIP súboru
- parameters - vnorený slovník, ktorý obsahuje parametre pripínania trás k cestnej sieti.

Po načítaní nastavení skriptu prebehne inicializácia pomocných premenných, do ktorých sa bude ukladať priebeh pripnutia trás k cestnej sieti. Inicializuje sa *successful* premenná, ktorá reprezentuje pole, do ktorého sa budú ukladať názvy úspešne spracovaných trás. Taktiež sa inicializuje *failed* premenná, ktorá reprezentuje slovník, do ktorého sa budú ukladať názvy neúspešne spracovaných trás ako klúče a textový retazec informujúci chybu ktorá nastala ako hodnotu ku klúču.

Po inicializácii prebehne kontrola *unzipdir* nastavenia. Kontroluje sa, či zadaná cesta odkazuje na priečinok, ktorý obsahuje len priečinky. V prípade, že cesta odkazuje na priečinok, ktorý obsahuje súbor, algoritmus končí a výstupom algoritmu je slovník, ktorý obsahuje chybovú hlášku "**názov\_súboru** is not a directory, check the zip structure.", ktorá je neskôr zobrazená používateľovi. Ďalej sa kontrolujú názovy podpriečinkov, ktoré priečinok s cestou *unzipdir* obsahuje. Ak sa v priečinku nachádza priečinok s názvom iným ako *Walk* alebo *Drive*, algoritmus končí a výstupom je slovník, ktorý obsahuje chybovú hlášku "**názov\_podpriečinku** does not match the specified directory name 'Walk' or 'Drive', check the zip structure." Následne sa prechádzajú jednotlivé súbory (trasy) najprv v *Drive* priečinku, neskôr v *Walk* priečinku. Skript na základe mena súboru identifikuje, o aký typ súboru ide a podľa toho načíta body trasy do premennej. Skript dokáže načítať body z *csv*, *geojson* a *gpx* súboru. V prípade, že je súbor iného typu, do premennej *failed* sa uloží meno trasy ako klúč a chybová hláška "*Points couldn't be extracted.*" ako hodnota.

Po načítaní bodov do premennej vstupuje táto premenná do funkcie *map-match*. Do tejto funkcie vstupujú aj ďalšie argumenty, *container*, *parameters* a *costing*. Premenná *costing* je určuje typ dopravy a je nastavená na základe priečinka, z ktorého bola trasa načítaná. Ak bola trasa načítaná z priečinka *Walk*, ide o pešiu chôdzu a do premennej sa uloží hodnota *pedestrian*. V opačnom prípade sa do nej uloží hodnota *auto*, reprezentujúca jazdu autom. Premenná *parameters* reprezentuje slovník parametrov pripínania trás k cestnej sieti. Patria sem parametre, ktoré určujú presnosť GPS zariadenia v metroch, ktorým bola trasa meraná, dĺžku polomeru kruhu, v ktorom sa majú hľadať kandidáti cestnej siete, ku ktorým sa má bod v trase pripnúť, taktiež v metroch a penalizácia odbáčania na druhú cestu určená celočíselnou hodnotou. Vo funkcií sa naformátujú body spolu s parametrami a druhom dopravy do textového refazca, ktorý je vložený do požiadavky na *Valhalla* kontajner. Ukážku textového refazca je možné vidieť vo výpise 1.

```
{
  "shape": [{"lat": 47.993351,"lon": 18.174553}, {"lat": 47.993351,"lon": 18.174553}],
  "shape_match": "map_snap",
  "costing": "auto",
  "costing_options": {"pedestrian": {"ignore_access": true}},
  "format": "osrm",
  "trace_options": {
    "search_radius": 50,
    "turn_penalty_factor": 200,
    "gps_accuracy": 5
  }
}
```

Listing 1: Dáta požiadavky na *Valhalla* kontajner

Je možné vidieť jednotlivé parametre pripínania a ostatné nastavenia. Nastavenie *ignore\_access* pre pešiu chôdzu nastavujeme, aby sme algoritmu oznámili, že má ignorovať typy ciest pri hľadaní kandidátov cestnej siete, ku ktorým má trasu pripnúť. Vďaka tomuto nastaveniu môže algoritmus pripnúť pešiu chôdzu aj na cyklotrasu.

Po odoslaní požiadavky na *Valhalla* kontajner a prijatí odpovede funkcia vráti odpoveď, ktorá je neskôr spracovaná. Z ukážky 2 možno vidieť, že trasa je zložená z rôznych segmentov, označenými atribútom *legs*. V ukážke 2 je zobrazený len jeden segment, aby výpis nebol zbytočne veľký. Je možné vidieť, že geometria nájdenej trasy obsahuje geometriu z prvého segmentu v *legs* atribúte.

```
{
  {
    "matchings": [
      {
        "weight_name": "pedestrian", "weight": 10027.532, "duration": 7257.535, "distance": 10239.001,
        "legs": [
          {
            "via_waypoints": [], "admins": [{}], "weight": 10027.532, "duration": 7257.535,
            "steps": [
              {
                "order": 1, "x": 18.174553, "y": 47.993351, "type": "walk"
              },
              {
                "order": 2, "x": 18.174553, "y": 47.993351, "type": "walk"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```

        "intersections": [
            { "bearings": [ 253 ], "entry": [ true ], "admin_index": 0, "out": 0,
              "geometry_index": 0, "location": [ 17.063915, 48.158126 ] }
        ],
        "maneuver": {
            "instruction": "Walk west.",
            "type": "depart",
            "bearing_after": 253,
            "bearing_before": 0,
            "location": [ 17.063915, 48.158126 ]
        },
        "name": "",
        "duration": 4.941,
        "distance": 7,
        "driving_side": "left",
        "weight": 4.941,
        "mode": "walking",
        "geometry": "{yizzAu}np_@b@rD"
    }],
    "distance": 10239.001,
    "summary": "Stare Grunty, Dvorakovo nabrezie"
}
],
"geometry":
    "{yizzAu}np_@b@rDxHaD~GsGjHiH|BsBpAdAgFnUs@~CzIoClp@mSni@}0j\\kJt@UltBun@jLgD|\\gKnKaG~QyKrBaA~EoCjK{GfNgKzLcK~AcC~CqFxc@{j@jIoKrB}CjGiJxt@q'AllKjLbCtSlCbEeCtDiGr@}J~@_b@NcHRyIhByz@|@e`@f@)T VcJhL^hJl@tETLlh@d@mSt@DcA|c@GdA_AYq@GhAmd@f@qJh@ag@z@g`pAiQjCaQbAaEhAeDbEkKdBgEhCkEhByBdDoC'Ri MnAg@bEk@`@UTq@jBmFvBiGXYjAgAfPuArBn@hSvFdYbIjFtAjh@vN|InCdGh@NiEd@gNeAGD{A@]Bu@HuBhAJGvBEpAw@I DqADwAr@)Wf~Hny@jw@fHla@tBBg@]1L1@F_BkYmBiEu}sHio@{CcWcCcWkAkQgA{WYyNO{N@yMhsM\`s0t@qQr@yL`A}L dCgV|C_VzFgc@bs@qsF|m@omGxs@ynHbC_WjAoZVqSEkS_AeZkBkZeD{W_Gq]}BuN_BoNaAeRUG0~AexAJuKnA{jAm@wSgB qNiCyNgGiSmAkHQoCYeEqIsmBMkCo@wMh@A@CnEK?}@egEpHs^gAq@CAeLj@?x@ACuKAgCbJYnC?bA?r@?v@?p@?hA?rC? fAhJDzTFzPBrNgC|zA}AjrA_BxrA_|@}Iht@kIrw@", "confidence": 1
}
],
"tracepoints": [
{
    "matchings_index": 0, "waypoint_index": 0, "alternatives_count": 0,
    "distance": 43.15, "name": "", "location": [ 17.063915, 48.158126 ]
}, { ... }
],
"code": "Ok"
}
}

```

Listing 2: Odpoved z *Valhalla* kontajnera

Z odpovede získame atribút *geometry*, ktorý reprezentuje zakódovanú geometriu cestnej siete, po ktorej sa predpokladá, že trasa prechádzala. Táto geometria sa dekóduje na body, ktoré sú uložené do súboru s názvom *map-match.csv*. Pôvodné body sú uložené do súboru s názvom *original.csv*. Tieto dva súbory sú potom uložené do priečinka *routes* pre príslušného používateľa, príslušný nahraný súbor a príslušnú trasu. Viac o štruktúre

*routes* priečinka je popísané v kapitole 4.2.

Nakoniec sa inicializuje premenná *retDict*, ktorá bude reprezentovať slovník. Príklad *retDict* premennej vidno na výpise 3, obsahuje nasledujúce kľúč-hodnota páry:

- failed - počet neúspešne spracovaných trás
- successful - počet úspešne spracovaných trás
- failed-info - premenná *failed*, ktorá obsahuje názov trasy ako kľúč a textový retazec s informáciou o chybe, ktorá nastala ako hodnotu

Výstupom skriptu je premenná *retDict* konvertovaná na textový retazec. Informácie z tohto slovníka sú neskôr zobrazené používateľovi, aby v prípade neúspešného pripnutia trás dostať informáciu o chybe a chybnej trase.

```
{  
  {  
    "failed": 3,  
    "successful": 519,  
    "failed_info": {  
      "u16112023_187923-189171_2024-01-13200431": "b'{\"code\": \"NoRoute\", \"message\": \"Impossible  
route between points\"}",  
      "u16112023_74346-75016_2023-12-09121131": "b'{\"code\": \"NoSegment\", \"message\": \"One of the  
supplied input coordinates could not snap to street segment.\"}",  
      "u16112023_85191-85958_2023-12-09151521": "b'{\"code\": \"NoSegment\", \"message\": \"One of the  
supplied input coordinates could not snap to street segment.\"}"  
    }  
  }  
}
```

Listing 3: *retDict* premenná

## **6 Zhodnotenie**

# Záver

Conclusion is going to be where?

Here.

# Zoznam použitej literatúry

1. *How to Build a Web App in 12 Simple Steps* [online]. [cit. 2024-03-30]. Dostupné z : <https://kissflow.com/application-development/how-to-create-a-web-application/>.
2. *What is the role of JavaScript in web application programming?* [online]. [cit. 2024-04-06]. Dostupné z : <https://www.linkedin.com/advice/0/what-role-javascript-web-application-programming-rb4kc>.
3. *What is MySQL?* [online]. [cit. 2024-04-06]. Dostupné z : <https://www.oracle.com/mysql/what-is-mysql/>.
4. *Mapbox* [online]. [cit. 2024-04-01]. Dostupné z : <https://github.com/mapbox>.
5. *Mapbox GL JS* [online]. [cit. 2024-04-01]. Dostupné z : <https://www.mapbox.com/mapbox-gljs>.
6. *Navigation SDK for mobile* [online]. [cit. 2024-04-01]. Dostupné z : <https://www.mapbox.com/navigation-mobile>.
7. *MapGPT* [online]. [cit. 2024-04-01]. Dostupné z : <https://www.mapbox.com/mapgpt>.
8. *GPS Visualizer: Do-It-Yourself Mapping* [online]. [cit. 2024-04-05]. Dostupné z : <https://www.gpsvisualizer.com/>.
9. *What is Docker?* [online]. [cit. 2024-01-04]. Dostupné z : <https://aws.amazon.com/docker/>.
10. *What is Docker and what are its advantages?* [online]. [cit. 2024-01-04]. Dostupné z : <https://www.dimensiona.com/en/what-is-docker-and-what-are-its-advantages/>.
11. [online]. [cit. 2024-01-04]. Dostupné z : <https://valhalla.github.io/valhalla/>.
12. [online]. [cit. 2024-01-04]. Dostupné z : <https://valhalla.github.io/valhalla/odin/>.
13. *Make an impact with your location data* [online]. [cit. 2024-01-04]. Dostupné z : <https://kepler.gl/>.
14. *Visualizing Data on a map has never been easier Kepler.GL* [online]. [cit. 2024-01-04]. Dostupné z : <https://www.linkedin.com/pulse/visualizing-data-map-has-never-been-easier-keplergl-ali>.

15. *From Beautiful Maps to Actionable Insights: Introducing kepler.gl, Uber's Open Source Geospatial Toolbox* [online]. [cit. 2024-01-04]. Dostupné z : <https://www.uber.com/en-SK/blog/keplergl/>.
16. *Express: Fast, unopinionated, minimalist web framework for Node.js* [online]. [cit. 2024-04-01]. Dostupné z : <https://expressjs.com/>.
17. [online]. [cit. 2024-04-01]. Dostupné z : <https://github.com/expressjs/express/releases>.
18. *Express.js: The Good, the Bad, and the Ugly* [online]. [cit. 2024-04-01]. Dostupné z : <https://www.linkedin.com/pulse/expressjs-good-bad-ugly-aziz-taha/>.
19. [online]. [cit. 2024-04-01]. Dostupné z : <https://code.visualstudio.com/docs>.
20. *Announcing Visual Studio Code - Preview* [online]. [cit. 2024-04-01]. Dostupné z : <https://web.archive.org/web/20151009211114/http://blogs.msdn.com/b/vscode/archive/2015/04/29/announcing-visual-studio-code-preview.aspx>.
21. *Why did we build Visual Studio Code?* [online]. [cit. 2024-04-01]. Dostupné z : <https://code.visualstudio.com/docs/editor/whyvscode>.
22. *Leaflet: an open-source JavaScript library for mobile-friendly interactive maps* [online]. [cit. 2024-04-15]. Dostupné z : <https://leafletjs.com/>.
23. *Leaflet vs OpenLayers. What to choose?* [online]. [cit. 2024-04-15]. Dostupné z : <https://www.geoapify.com/leaflet-vs-openlayers>.
24. *Leaflet Quick Start Guide* [online]. [cit. 2024-04-15]. Dostupné z : <https://leafletjs.com/examples/quick-start/>.