

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

DYNAMICKÁ NAVIGÁCIA OSÔB SO SPÄTNOU
VÄZBOU V MESTSKEJ HROMADNEJ DOPRAVE
BAKALÁRSKA PRÁCA

2016
JAKUB NOVÁK

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

DYNAMICKÁ NAVIGÁCIA OSÔB SO SPÄTNOU
VÄZBOU V MESTSKEJ HROMADNEJ DOPRAVE
BAKALÁRSKA PRÁCA

Študijný program: Informatika
Študijný odbor: 2508 Informatika
Školiace pracovisko: Katedra informatiky
Školiteľ: RNDr. Marek Nagy, PhD.

Bratislava, 2016
Jakub Novák



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Jakub Novák
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Dynamická navigácia osôb so spätnou väzbou v mestskej hromadnej doprave
Dynamic navigation of persons with feedback for public transport

Cieľ: Vytvorenie webovej aplikácie, ktorá zjednoduší plánovanie cesty dopravnými prostriedkami MHD. Využitie budú online mapy a informácie o trasách a odchodoch liniek MHD. Aplikácia naplánuje a vyznačí optimálnu trasu, v ktorej zohľadní aj čas pešieho presunu. Na základe spätnej väzby užívateľov aplikácia vykoná časové korekcie. Napr. podľa zmeny GPS polohy zráta priemerný čas presunu, podľa rýchlosti zmeny a pohybu GPS odhadne, či užívateľ cestuje danou linkou MHD, ...

Vedúci: RNDr. Marek Nagy, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.
Dátum zadania: 25.10.2015

Dátum schválenia: 29.10.2015

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

PodĎakovanie: Chcel by som sa poĎakovať môjmu školiteľovi, RNDr. Marekovi Nagymu, PhD., za jeho vedenie a pripomienky k mojej práci. Taktiež ďakujem Petrovi Poláčikovi, Marekovi Šuppovi za ich pomoc, nápady a trpezlivosť a všetkým ostatným, ktorí ma podporili a poskytli cenné rady.

Abstrakt

Bakalárska práca sa zaoberá implementáciou webovej aplikácie, ktorá slúži na zjednotenie pohybu po Bratislave pomocou mestskej hromadnej dopravy. Práca porovnáva poskytovateľov mapových služieb, popisuje získavanie potrebných údajov, architektúru systému, použité knižnice a okrem samotnej implementácie aj problémy, ktoré nastali pri implementácii. Aplikácia je schopná vyhľadávať trasy na základe zadanej fyzickej adresy, ale aj lokácie zariadenia, nájsť najbližšiu zastávku, vyhľadať spoje a navigovať používateľa až k cieľovému bodu.

Kľúčové slová: webová aplikácia, open-source, navigácia, python, mhd

Abstract

The bachelor's thesis is concerned with implementation of web application which simplifies movement around Bratislava via public transport. The thesis compares different map services, explains retrieval of needed data, system architecture, used libraries, implementation and problems. Application is able to find routes when provided with an address, as well as the location of the device, find the nearest stop, appropriate public transport and navigate user to the end point.

Keywords: web application, open-source, navigation, python, public transport

Obsah

Úvod	1
1 Podobné projekty	2
1.1 Imhd	2
1.2 9292	2
2 Informácie	3
3 Porovnanie poskytovateľov máp	5
3.1 OpenStreetMap	5
3.1.1 Formát dát	5
3.1.2 Application programming interface	6
3.1.3 Licencia	7
3.2 Google Maps	7
3.2.1 Formát dát a API	7
3.2.2 Licencia	9
3.3 Bing Maps	10
3.3.1 Formát dát a API	10
3.3.2 Licencia	11
3.4 Zvolené riešenie	11
4 Implementácia	12
4.1 Programovacie jazyky	12

4.1.1	Python	12
4.1.2	JavaScript	14
4.2	Architektúra systému	15
4.3	Geopy	16
4.3.1	Geocoding	17
4.3.2	Distance	17
4.4	Mapbox	18
4.4.1	Directions	18
4.4.2	Distance	19
4.5	Imhdk	21
4.6	Backend	21
4.6.1	normalize	21
4.6.2	get_coordinates a get_address	21
4.6.3	get_distance_meters a get_distance_seconds	21
4.6.4	get_midpoint	21
4.6.5	get_stop_name a get_stop_location	22
4.6.6	check_preferred_way	22
4.6.7	identify_line	22
4.6.8	get_mhd	23
4.6.9	find	23
4.7	Frontend	25
4.7.1	HTML5 Geolocation	25
4.7.2	Používateľské rozhranie	26
5	Rozšírenia	28
5.1	Zastávky spojov	28
5.2	Používateľská odozva	28
5.3	Výluky	29
5.4	Sledovanie zariadenia navigovaného používateľa	29

<i>OBSAH</i>	viii
5.5 Rozšírenie o ďalšie mestá	30
5.6 Chat a zdieľanie	30
5.7 Pomoc nevidiacim	30
6 Testy	31
7 Inštalácia a spustenie	32
Záver	34
Dodatok A	37

Zoznam obrázkov

2.1	Stredný bod	4
4.1	Diagram architektúry systému	15
4.2	Usecase diagram používateľa	16
4.3	Hlavná stránka	26
4.4	Stránka s výsledkami	27

Úvod

V Bratislave, aj kdekoľvek na Slovensku je problematické sa navigovať využívajúc MHD, ak dané miesto nepoznáme. Ak napríklad nepoznáme presné názvy zastávok MHD, ktoré často nezodpovedajú názvu ulice, na ktorej sa nachádzajú, stáva sa z navigácie po neznámom meste veľmi nepríjemná skúsenosť. Preto sme sa rozhodli vytvoriť aplikáciu, ktorá tento problém pomôže vyriešiť. Predstavme si nasledovnú situáciu: Sme v Bratislave a chceme sa dostať z miesta, kde sa nachádzame (A) na miesto (B), kde sa nachádza náš priateľ, ktorý sa tiež v Bratislave až tak dobre nevyzná. Obrátime sa teda na našu aplikáciu. Ak máme zapnuté GPS, stačí zadať adresu, kde sa nachádza kamarát, ak GPS zapnuté nemáme, zadáme ručne aj adresu, na ktorej sa nachádzame. Aplikácia potom vyhodnotí najlepšiu a najrýchlejšiu cestu z bodu A do bodu B. Na mape nám ukáže trasu na najbližšiu zastávku, na ktorý spoj treba nastúpiť, kedy vystúpiť, prípadné presuny medzi zastávkami ukáže na mape, a nakoniec cestu zo zastávky k nášmu cieľu, bodu B.

Cieľom tejto bakalárskej práce je zoznámiť sa s programovacím jazykom Python a pomocou neho vyvinúť webovú aplikáciu, ktorá naplánuje optimálnu trasu, v ktorej zohľadní aj čas pešieho presunu. Táto aplikácia potom na mape vyznačí najkratšiu trasu aj s časmi. Popíšeme aj možné naimplementovanie spätnej väzby od používateľov, na základe ktorej by sme mohli vykonávať časové korekcie.

Písomná časť sa skladá zo 7 kapitol - podobné projekty, informácie, porovnanie poskytovateľov máp, implementácia, rozšírenia, testy, inštalácia a spustenie. Kapitola 1 popisuje súvisiace práce, ktorými sme sa inšpirovali. Kapitola 2 sa venuje zdrojom dát, ktoré sme získavali. Kapitola 3 porovnáva poskytovateľov máp, popisuje výhody a nevýhody, obmedzenia a API poskytovateľov. Kapitola 4 sa týka samotnej implementácie webovej aplikácie. Popisujeme výber programovacieho jazyka, Kapitola frameworku, architektúru systému, použité knižnice, hlavné časti aplikácie a užívateľské prostredie. Kapitola 5 hovorí o možných rozšíreniach našej aplikácie. Kapitola 6 ukazuje rýchlosť vyhľadania trasy v závislosti od rôznych faktorov. Kapitola 7 naviguje používateľa inštaláciou našej aplikácie.

Kapitola 1

Podobné projekty

V tejto kapitole popíšeme projekty, ktoré sa zaoberajú podobnou problematikou.

1.1 Imhd

Stránky imhd.sk vznikli v roku 2000 ako nekomerčný projekt zameraný na podporu mestskej hromadnej dopravy v Bratislave a postupom času sa rozšírili aj o ďalšie slovenské mestá. Imhd umožňuje vyhľadať spojenie medzi dvomi bodmi, rovnako umožňuje aj pohodlný prehľad cestovných poriadkov jednotlivých liniek mestskej hromadnej dopravy. Linky, trasy aj zastávky je možné pre zjednodušenie vyhľadávania si uložiť ako svoje obľúbené. Cestovné poriadky si môžete pozrieť aj v offline režime. Imhd spolupracuje s dopravným podnikom, odkiaľ čerpajú dáta o spojoch, zastávkach, atď.

1.2 9292

9292.nl je plánovač trasy s podporou všetkej hromadnej dopravy v Holandsku. Ponúkajú informácie o trase v reálnom čase, viac možností vybraní štartovacieho bodu a finálneho bodu - pomocou aktuálnej GPS lokácie, autobusovej zastávky, adresy, atď. Používateľ si môže vytvoriť profil, kde sa mu budú ukladať často používané lokácie, aby ich nemusel neustále zadávať, ktoré sú synchronizované so všetkými zariadeniami. Aplikácia upozorňuje aj na nečakané zmeny v trase, aby používateľa dostala do destinácie bez akýchkoľvek meškaní. Projekt vznikol v roku 1991 začatím spolupráce všetkých inštitúcií, ktoré sa zaoberali mestskou hromadnou dopravou.

Kapitola 2

Informácie

Táto kapitola sa zaoberá hľadaním potrebných zdrojov a zastávok MHD.

Momentálne neexistuje jednoduchý spôsob, ako by sme mohli získať zoznam zastávok s ich príslušnými súradnicami. Jedna z možností je ručné extrahovanie zastávok zo zdroja, ako napríklad Google Maps.

Neskôr sa nám podarilo nájsť databázu OpenStreetMap, kde sa nachádzali všetky zastávky, ale ich získanie sa tiež ukázalo zložitejšie, než sa na prvý pohľad zdalo. Bolo nutné nájsť jeden konkrétny element v zdrojovom kóde, ktorý obsahoval časť url, ktorú následne bolo potrebné doplniť do jej druhej časti, aby sme sa dostali ku zdrojovému kódu, v ktorom sa konečne nachádzali súradnice hľadanej zastávky skryté medzi ďalšími, pre nás nepodstatnými, informáciami. Automatizácia tohto procesu by bola zbytočne náročná, tak sme sa rozhodli hľadať iný zdroj, než sa pustíme do extrakcie.

Freemap.sk je voľne dostupná mapa, využívajúca OpenStreetMap, ktorá ponúka možnosť exportovať zobrazené objekty, medzi ktorými sa nachádzajú aj električkové a autobusové zastávky v zobrazenom výreze mapy. Hlavný problém tohto prístupu je limitácia freemap.sk, ktorá nedovoľuje zobraziť na mape viac ako 1000 objektov naraz. Preto sme rozdelili Bratislavu na menšie štvorce, z ktorých každý neprekračoval tento limit a obsahoval menej ako 1000 objektov. Tieto štvorce sme následovne spojili do jedného súboru. Výsledný súbor obsahoval všetky hľadané informácie o zastávkach.

```
<?xml version='1.0' encoding='UTF-8' ?><gpx version="1.1" creator="Freemap  
Slovakia, CC-BY-SA 2.0" xmlns="http://www.topografix.com/GPX/1/1" ><wpt  
lat="48.1767134" lon="17.1465039" ><name>Autobusova zastavka Odborarska</  
name></wpt><wpt lat="48.1879351" lon="17.0363863" ><name>Autobusova zastavka
```

```

Drobneho</name></wpt><wpt lat="48.1859181" lon="17.053363" ><name>Autobusova
zastavka Segnare</name></wpt><wpt lat="48.1825201" lon="17.1713235" ><name>
Autobusova zastavka Shopping Park Soravia</name></wpt>

<wpt lat="48.1857811" lon="17.1331615" ><name>Zastavka elektricky ZST Vinohrady
</name></wpt><wpt lat="48.1767018" lon="17.1462562" ><name>Zastavka elektricky
Odborarska</name></wpt>

```

Listing 1: Ukážka z XML súboru so zastávkami

Pomocou knižnice `xml.dom.minidom` sme následne vytvorili skript na extrahovanie názvov a súradníc zastávok, ktoré sme si vložili do python slovníka pre jednoduchú manipuláciu s týmito dátami.

```

1 trams = {
2     'STU': [(48.1523201, 17.1157568), (48.1512188, 17.1160848)],
3     'Namestie SNP': [(48.1451993, 17.1116367)],
4     'Kamenne nam.': [(48.1455395, 17.113559)],
5     ...
6 }

```

Listing 2: Ukážka zo slovníka s električkovými zastávkami

Keďže momentálne nie je možné zistiť, ktorým smerom ide spoj a ktorá konkrétna zastávka patrí ktorému spoju, definujeme na mape stredný bod. Tento bod potom slúži ako všeobecný bod pre zastávku s daným názvom a taktiež aj ako bod, okolo ktorého je mapa neskôr vycentrovaná.

Stredný bod hľadáme spočítaním všetkých zemepisných výšok a širok a ich predeľením počtom všetkých súradníc.

$$\left(\frac{P_1^{latitude} + P_2^{latitude} + \dots + P_n^{latitude}}{n}, \frac{P_1^{longitude} + P_2^{longitude} + \dots + P_n^{longitude}}{n} \right)$$

Obr. 2.1: Stredný bod

Pre zjednodušenie vytvorenia stredného bodu je kľúčom slovníka meno zastávky (kvôli vyhľadaniu) a jeho hodnota je pole súradníc, ktorým prislúcha rovnaký kľúč.

Kapitola 3

Porovnanie poskytovateľov máp

V tejto kapitole popíšeme OpenStreetMap, Google Maps, Bing Maps a ich porovnaním.

3.1 OpenStreetMap

OpenStreetMap (ďalej len OSM) je bezplatná, editovateľná mapa celého sveta, ktorá je udržiavaná dobrovoľníkmi, ktorí sa rozhodli prispieť k vývoju vo svojom voľnom čase, napríklad reálnym zbieraním dát v teréne pomocou GPS prístrojov. OSM licencia ponúka bezplatný prístup k mapovým obrázkom a všetkým ďalším potrebným dátam. OSM taktiež ponúka rôzne vrstvy zobrazenia, ako napríklad mapa pre cyklistov, transportná mapa, atď [1]. OSM poskytuje API (application program interface) na spracovanie dát z ich databázy. Okrem bežných užívateľov je tento projekt podporovaný mnohými open-source projektami, ktoré prispievajú ku zvýšeniu kvality dát a rýchlosti aktualizácií.

3.1.1 Formát dát

V OSM databáze sa nachádzajú dáta tohto typu:

- Uzly (nodes) - súradnice, alebo skupina súradníc. Každá súradnica je definovaná zemepisnou šírkou a dĺžkou.
- Trasy (ways) - postupnosť súradníc, alebo kolekcia týchto postupností. Každá trasa je tvorená najmenej dvoma uzlami. OSM vie pomocou trás reprezentovať

aj uzavrené postupnosti súradníc (mnohouholníky). Tieto trasy majú rovnaký začiatočný a koncový uzol.

- Relácie (relations) - skupina uzlov, trás, prípadne ďalších relácií. Relácia má za úlohu vytvoriť prepojenia medzi uzlami a trasami. Každá trasa/uzol v relácii má priradené atribúty.
- Atribúty (attributes) - vlastnosti uzlov, trás a relácií. Napríklad hovoria, či je daná cesta pešia, cyklistická, alebo pre autá.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2" >
  <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900"
    maxlon="12.2524800"/>
  <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHR0"
    uid="46882" visible="true" version="1" changeset="676636"
    timestamp="2008-09-21T21:37:45Z"/>
  <way id="26659127" user="Masch" uid="55988" visible="true" version="5"
    changeset="4142606" timestamp="2010-03-16T11:47:08Z" >
    <nd ref="292403538"/>
    <tag k="highway" v="unclassified"/>
    <tag k="name" v="Pastower Stra\T1\ss e"/>
  </way>
  <relation id="56688" user="kmvar" uid="56190" visible="true" version="28"
    changeset="6947637" timestamp="2011-01-12T14:23:49Z" >
    <member type="node" ref="294942404" role=""/>
    <member type="way" ref="4579143" role=""/>
  </relation>
</osm>
```

Listing 3: Skrátená ukážka OSM XML súboru

3.1.2 Application programming interface

Existuje viacero Python knižníc, ktoré pracujú s OSM API [2].

- PyrouteLib - knižnica na hľadanie optimálnej trasy v mestách, ktorá používa vyhľadávací algoritmus A*. Medzi hlavné výhody patrí absencia potreby databázového servera k jej činnosti, je vhodná na vyhľadávanie trás v mestách a je

optimalizovaná pre chodcov, cyklistov a autá. K jej nevýhodám patrí nemazanie dát uložených vo vyrovnávacej pamäti a neukladanie pozície na mape po reštarte/vypnutí aplikácie [3].

- SimpleOsmRouter - veľmi jednoduchý plánovač trás, bez ďalšej funkcionality [4].
- Overpass API - knižnica optimalizovaná na čítanie dát. Funguje ako databáza na internete: dotaz sa odošle do API a odpoveďou sú dáta, ktoré súhlasia s daným dotazom. Vďaka veľkej redukcii dostupných prvkov je optimalizovaná primárne pre čítanie prvkov databázy. Hlavnou nevýhodou je sťahovanie veľkých dát: veľkosť odpovede na dotaz je známa až po jeho úplnom stiahnutí [5].

3.1.3 Licencia

V roku 2012 prešli OSM z Creative Commons na Open Database Licence, čo znamená, že všetky dáta sú voľne prístupné na kopírovanie, upravovanie a distribúciu pre každú osobu, či firmu, ak je ako zdroj uvedený OSM a prispievatelia do tohto projektu. Ďalšou podmienkou je, že všetky dáta ktoré užívateľ vloží budú sprístupnené pre ostatných používateľov.

3.2 Google Maps

Google Maps je webová mapovacia služba od firmy Google, ktorá ponúka satelitné snímky, mapy ulíc, aktuálnu situáciu v doprave a plánovanie trás pre cestovanie pešo, autom, alebo mestskou hromadnou dopravou na vybraných miestach. Podobne ako OSM, aj Google Maps ponúka API, ktorá umožňuje vkladať mapy do webstránok tretej strany a ponúka vyhľadávač firiem a organizácií v mnohých krajinách [6].

3.2.1 Formát dát a API

Google Maps API sa skladá z viacerých rôznych API. Pre nás podstatné sú Google Directions (vytváranie trasy z bodu A do bodu B) a Google Geocoding (vyhľadávanie polohy podľa poskytnutej adresy/bodu na mape).

Google Directions API na dotaz odpovedá poľom trás (ak na dotaz neexistuje odpoveď, napríklad kvôli neexistujúcemu miestu, vráti prázdne pole) [7]. Každá trasa v poli obsahuje tieto dáta:

- Zhrnutie (summary) - krátky opis trasy.
- Vetvy (legs) - pole, ktoré obsahuje informácie o vetvách trasy.
- Poradie bodov trasy (waypoint_order) - pole indikujúce poradie bodov trasy.
- Autorské práva (copyright) - text s autorskými právami, ktoré je potrebné zobraziť pre danú trasu.
- Upozornenia (warnings) - pole s varovaniami, ktoré je potrebné zobraziť pre danú trasu.
- Cestovné (fare) - ak trasa obsahuje nejaké poplatky o ktorých Google vie. Cestovné obsahuje: Mena (currency), Cena (value) - celková cena trasy a Text (text) - celková cena trasy naformátovaná v požadovanom jazyku.

```
{
  "status": "OK",
  "geocoded_waypoints" : [
    {
      "geocoder_status" : "OK",
      "place_id" : "ChIJ7cv00DwsDogRAMDACA2m4K8",
      "types" : [ "locality", "political" ]
    },
  ],
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode": "DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },

```

Listing 4: Skrátená ukážka Directions API odpovede

Google Geocoding API ponúka informácie o polohe [8]. Skladá sa z dvoch podčastí: Geocoding a Reverse Geocoding. Geocoding je, všeobecne, preklad človekom

zadanej adresy na bod na mape. Reverse geocoding je opak geokódovania, teda vyhľadanie presnej adresy na základe aktuálnej polohy. Forma dát z Geocoding a Reverse Geocoding je rovnaká. Niektoré z vrátených dát sú napríklad Adresa (street_address) - presná adresa ulice, Štát (country) - názov štátu v ktorom sa zariadenie nachádza, Letisko, park, susedstvo, atď.

```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "1600",
          "short_name" : "1600",
          "types" : [ "street_number" ]
        },
        {
          "long_name" : "United States",
          "short_name" : "US",
          "types" : [ "country", "political" ]
        },
      ],
      "formatted_address" : "1600 Amphitheatre Parkway, Mountain View,
                           94043, USA",
      "geometry" : {
        "location" : {
          "lat" : 37.4224764,
          "lng" : -122.0842499
        }
      }
    }
  ]
}
```

Listing 5: Skrátená ukážka Geocoding API odpovede

3.2.2 Licencia

Google Maps je uzatvorený systém. Všetky vložené informácie sa okamžite stávajú majetkom Google bez akýchkoľvek nárokov na odmenu. Podobne aj všetky dáta, ktoré akýmkoľvek spôsobom použité, sú pod autorskými právami Google a spoločností, od ktorých Google tieto dáta odkúpil [9]. Ďalší problém je strata súkromia a kontroly

nad tým, čo vidíme na mape. V štandardnej bezplatnej verzii poskytujú služby Google Directions aj Google Geocoding najviac 2500 požiadavok za deň, pričom vrámci Google Direction jedna požiadavka pozostáva z najviac 10 navigačných bodov [10].

3.3 Bing Maps

Bing maps je mapovacia služba od firmy Microsoft. Ponúka mnoho vrstiev dát nad licencovanými mapami a možnosť tieto vrstvy ľubovoľne prekryvať podľa potreby. Dostupné sú satelitné snímky, letecké snímky, street view, 3D modely miest a terénu a vtáčie oko (snímky zachytené nízko letiacim lietadlom) [11].

3.3.1 Formát dát a API

Trasa Bing Maps má nasledovnú základnú štruktúru [12].

- Trasa (Route)
 - Vetva trasy (RouteLeg) - časť trasy medzi počiatočným a koncovým bodom.
 - * Podvetva trasy (RouteSubLeg) - časť trasy medzi počiatočným/koncovým bodom a nejakým bodom, ktorý sa nachádza medzi nimi, alebo medzi dvoma bodmi, ktoré sa nachádzajú medzi počiatočným a koncovým bodom.
 - * Prvok trasy (ItineraryItem)
 - Detaily (Detail)
- Cesta trasy (RoutePath)

Bing Maps sa skladá z viacerých rôznych API. Pre nás podstatné sú Routes (vytváranie trasy z bodu A do bodu B) a Locations (vyhľadávanie polohy podľa poskytnutej adresy/bodu na mape).

Niektoré z vrátených dát po odoslaní dotazu na Routes API sú napríklad dĺžka trasy (travelDistance), čas potrebný na prekonanie trasy (travelDuration) a cesta trasy (routePath) [12].

Niektoré z vrátených dát po odoslaní dotazu na Locations API [13]:

- Bod (Point) - súradnice bodu na mape.

- Adresa (Address) - poštová adresa bodu na mape. Môže obsahovať aj ďalšie údaje o bode na mape, ako napríklad: susedstvo, poštové smerové číslo, región krajiny, atď.
- Dôveryhodnosť (Confidence) - ukazuje, či prijatá adresa súhlasí so zadanou.

3.3.2 Licencia

Podobne ako Google Maps, aj Bing Maps sú uzatvorený systém a všetky vložené dáta patria Microsoft-u a ich licenčným partnerom. Microsoft ponúka špeciálnu licenciu pre vzdelávacie organizácie, avšak mnohé funkcie sú pod touto bezplatnou licenciou obmedzené a dostupné len pre komerčnú licenciu [14].

3.4 Zvolené riešenie

Prvotne sme pre našu aplikáciu plánovali použiť Google Maps, prípadne Bing Maps (ak by poskytovali nejaké výhody), avšak po dôkladnejšom zvážení sme sa rozhodli použiť OpenStreetMaps. OpenStreetMaps majú za sebou veľmi aktívnu komunitu, takže sú pravidelne aktualizované. Ďalej všetky dáta sú pod licenciou pre voľné používanie bez akýchkoľvek limitácií a existuje mnoho rôznych API, z ktorých si môžeme vybrať tú najvhodnejšiu pre našu aplikáciu.

Kapitola 4

Implementácia

V tejto kapitole popíšeme vybraný programovací jazyk, porovnáme rôzne vhodné frameworky, knižnice a implementáciu. V implementácii sme sa snažili o čo najlepšiu rozširiteľnosť a čitateľnosť kódu. Celý náš projekt je dostupný na stránke <https://github.com/JakubNvk/bc> a vydaný pod štandardnou open-source licenciou.

4.1 Programovacie jazyky

Keďže chceme vytvoriť webovú aplikáciu, je vhodné vybrať programovací jazyk, pre ktorý existuje kvalitný framework na prácu s webom.

4.1.1 Python

Python je dynamický, vysoko-úrovňový programovací jazyk. Je to jednoduchý, interpretovaný jazyk s minimalistickou syntaxou, ktorá používa odrážky na oddelenie blokov kódu, na rozdiel ostatných rozšírených jazykov, ktoré používajú zátvorky. Python je open-source projekt, dostupný pre všetky bežné platformy a je zahrnutý vo väčšine Linuxových distribúcií. Má jednoduchú a prehľadnú syntax a podporuje viacero programovacích paradigmat, teda umožňuje programátorom využívať objektovo-orientované, procedurálne, alebo funkcionálne techniky prístupu.

Python slúži aj ako skriptovací jazyk pre webové aplikácie a ponúka mnoho frameworkov ako napríklad Django, Flask, Bottle, atď.

Django

Django je najpoužívanejší framework, čo sa týka tvorby webových aplikácií na pythonovskej báze. Nasleduje filozofiu `batteries included`, čo znamená, že ponúka širokú funkcionality bez nutnosti sťahovania ďalších dodatkov a preto je najvhodnejší na prácu s väčšími projektami, avšak je dobrý aj na prácu s menšími aplikáciami v pythone.

```
hello_django
  /hello_django
    /__init__.py
    /settings.py
    /urls.py
    /wsgi.py
  /howdy
    /admin.py
    /__init__.py
    /migrations
      /__init__.py
    /models.py
    /tests.py
    /views.py
  /manage.py
```

Listing 6: Django - súborová štruktúra

Flask

Flask je "microframework", čo znamená, že je primárne vhodný pre tvorbu menších projektov, ktoré potrebujú rýchly spôsob ako vytvoriť pythonovskú webovú aplikáciu a ponúka jednoduché rozšírenie vďaka mnoho dostupným pluginom. Na Flask frameworku je postavených mnoho jednoduchých web stránok. Z Flasku často benefitujú backend projekty, ktoré potrebujú jednoduché, rýchle a ľahko konfigurovateľné webové rozhranie.

```
/application
  /__init__.py
  /templates
    /hello.html
```

Listing 7: Flask - súborová štruktúra

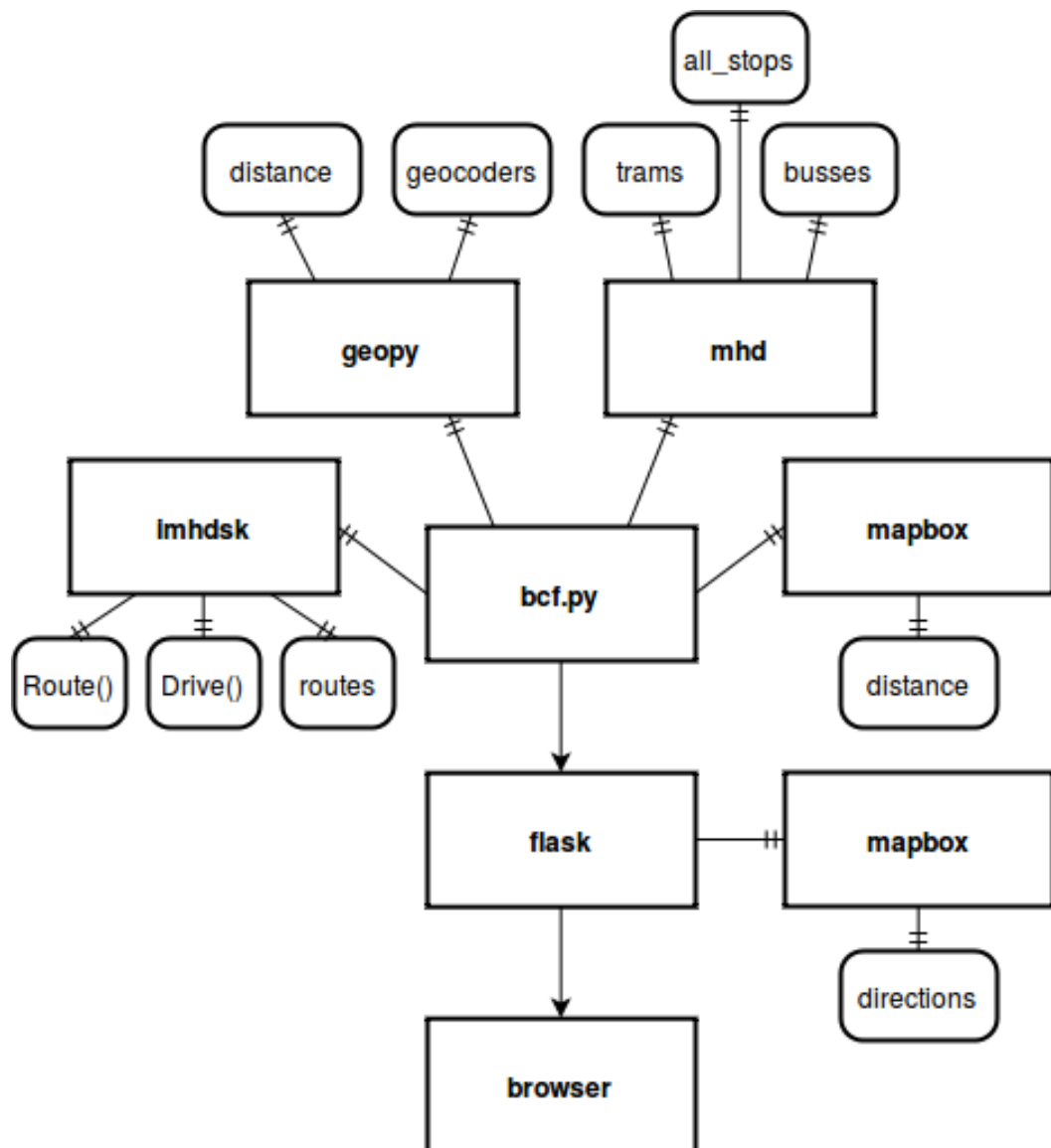
Zvolené riešenie

Flask je často odporúčaný pre naučenie sa základov tvorby webových aplikácií, lebo pomáha pochopiť tvorbu webu vďaka jeho jednoduchej štruktúre. Keďže sme pred týmto projektom nemali žiadne skúsenosti s tvorbou pythonových webových aplikácií, rozhodli sme sa použiť práve Flask.

4.1.2 JavaScript

JavaScript je dynamický, beztypový, interpretovaný programovací jazyk. V súčasnosti je pri HTML a CSS jeden z troch hlavných technológií pre tvorbu webu. Ako multi-paradigmaticý jazyk podporuje objektovo orientované, imperatívne a funkcionálne programovanie. JavaScript budeme používať pri zobrazovaní máp a ďalších inštrukcií o trase.

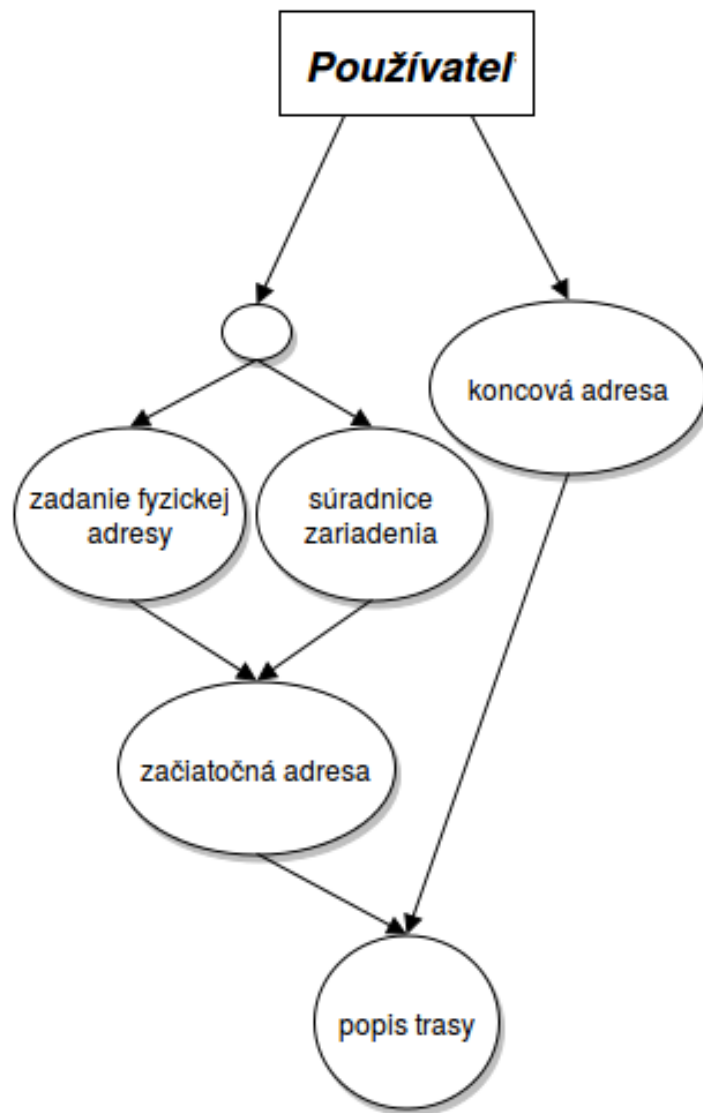
4.2 Architektúra systému



Obr. 4.1: Diagram architektúry systému

Hlavným súborom je pre nás **bcf.py**, v ktorom sme vytvorili všetky funkcie na spracovanie a prípravu údajov pre flask. Taktiež tu prebieha väčšina práce s API.

V súbore **app.py** sa nachádza implementácia serverovej strany našej webovej aplikácie, ktorá používa flask microframework. Táto aplikácia použije údaje z bcf.py, ktoré vloží do HTML súborov zobrazených používateľovi.



Obr. 4.2: Usecase diagram používateľa

Používateľ musí zadať 2 hodnoty - začiatočnú adresu a koncovú adresu. Pre začiatočnú adresu má možnosť automatickej lokácie svojho zariadenia, alebo môže adresu zadať ručne. Po zadaní oboch hodnôt a stlačení tlačidla pre vyhľadanie trasy sa mu zobrazí popis trasy.

4.3 Geopy

Geopy je Python klient pre internetové geocoding služby. Zjednodušuje získavanie súradníc adries, miest, atď. pomocou geocoderov tretej strany a iných zdrojov dát.

4.3.1 Geocoding

Na geocoding používame službu Geopy.geocoding. Geopy.geocoding vracia Location objekt, ktorý v sebe obsahuje adresu, zemepisnú výšku a šírku, nadmorskú výšku a neformátované dáta, ktoré vráti geocoder, v mojom prípade Nominatim. Nominatim je nástroj na prehľadávanie OSM dát podľa mena a adresy (geocoding) a na generovanie adries z OSM bodov (reverse geocoding).

```

1 def get_coordinates(address):
2     city = 'Bratislava'
3     location = Nominatim(timeout=10).geocode('{0}, {1}'.format(address, city))
4     if location is None:
5         return None
6     return (location.latitude, location.longitude)

```

Listing 8: Funkcia `get_coordinates`

4.3.2 Distance

Geopy vie vypočítať geodetickú vzdialenosť medzi dvoma bodmi pomocou formúl Vincenty a Great-circle.

- Great-circle (`great_circle`) používa sférický model Zeme s priemerom hlavnej kružnice 6372.795 kilometrov. To spôsobuje nepresnosť cca. 0.5%. Hodnota rádiusu je uložená v `distance.EARTH_RADIUS`, takže táto hodnota môže byť zmenená (avšak vždy by mala byť v kilometroch).
- Vincenty používa presnejší elipsoidový model Zeme. Existuje viac populárnych elipsoidových modelov. Najpresnejší model závisí od lokácie bodov na Zemi. Predvolený používaný model je elipsoid WSG-84, ktorý je všeobecne najpresnejší. Geopy obsahuje aj ďalšie modely v slovníku `distance.ELLIPSOIDS`.

Pre našu aplikáciu použijeme vincenty s modelom WSG-84.

	model	major (km)	minor (km)	flattening
ELLIPSOIDS = {'WSG-84':		(6378.137,	6356.7523142,	1 / 298.257223563),
	'GRS-80':	(6378.137,	6356.7523141,	1 / 298.257222101),

```

'Airy (1830)': (6377.563396, 6356.256909, 1 / 299.3249646),
'Intl 1924': (6378.388, 6356.911946, 1 / 297.0),
'Clarke (1880)': (6378.249145, 6356.51486955, 1 / 293.465),
'GRS-67': (6378.1600, 6356.774719, 1 / 298.25),
}

```

Listing 9: Slovník distance.ELLIPSOIDS

4.4 Mapbox

Mapbox je poskytovateľ prispôsobených online máp, z ktorých mnohé využívajú práve OpenStreetMap dáta a zapájajú sa do mnohých open-source mapovacích knižníc a aplikácií. Mapbox je hlavným prispievateľom do Mapnik projektu (nástroj na renderovanie máp). Pomohli vytvoriť Tilemill, softvér použitý na dizajn štýlu OSM máp. Mapbox zamestnáva tím špeciálne určený na vylepšovanie OSM.

Od Mapbox budeme využívať dve služby - Directions API a Distance API. Obe služby sú v beta prevádzke. Pre používanie Distance API bolo potrebné písomne požiadať o aktiváciu. Na túto požiadavku nám obratom vyhovelí a sprístupnili API.

4.4.1 Directions

Directions API umožňuje zobrazenie trasy z bodu A do bodu B na mape. Hľadá vždy najkratšiu cestu a ponúka tri plánovacie profily:

- mapbox.driving - profil pre plánovanie cesty pre autá a motocykle. Priorizuje diaľnice a podobné cesty, pre vyhľadanie najrýchlejšej trasy.
- mapbox.cycling - profil pre plánovanie cesty pre cyklistov. Priorizuje ulice, ktoré majú cyklistický pruh a snaží sa vyhnúť rýchlostným cestám.
- mapbox.walking - profil pre plánovanie cesty pre chodcov. Priorizuje chodníky.

API vracia textové inštrukcie, alternatívne trasy a geometriu pre vykresľovanie trás.

4.4.2 Distance

Distance API vracia JSON objekt, ktorý obsahuje maticu vzdialeností medzi všetkými prijatými lokáciami. Matica vzdialeností je vytvorená pomocou Directions API.

Z našej aplikácie budeme posielať požiadavku na vzdialenosť medzi dvoma bodmi, takže dostaneme maticu, ktorá vyzerá nasledovne:

```
{
  "durations": [
    ["A to A", "A to B"],
    ["B to A", "B to B"]
  ]
}
```

Listing 10: Matica vzdialeností

Zaujímá nás teda druhý prvok prvého poľa, alebo prvý prvok druhého poľa. Funkcia vracia dĺžku pešieho presunu z bodu A od bodu B v sekundách.

Táto API bola kľúčová pri hľadaní najbližšej zastávky. Keď sme používali Geopy na nájdenie najbližšej zastávky, v niektorých prípadoch funkcia vracala zastávku, ktorá síce bola bližšie geograficky, avšak peším presunom by používateľ v snahe dostať sa na danú zastávku prešiel okolo zastávky, ktorá bola reálne bližšie. Riešenie tohto problému ponúkla práve Distance API a po získaní prístupu k nej sme funkciu `get_distance_meters`, ktorú využívame pri hľadaní najbližšej zastávky, zmenili aby túto zastávku hľadala pomocou Distance.

[illegible]

```
13     response = distance_service.distances([loc_a, loc_b], 'walking')
14     return distance_seconds
```

Listing 11: Funkcia `get_distance_seconds`

To ale spôsobilo extrémne spomalenie aplikácie. Distance je omnoho pomalší než Geopy, lebo ráta reálnu pešiu vzdialenosť na rozdiel od vicenty, ktorý vyráta vzdialenosť geografickú. Preto sme sa rozhodli zakomponovať oboje API. Pomocou vincenty sme počet zastávok, pre ktoré hľadáme reálnu vzdialenosť, znížili vylúčením tých, ktoré sú mimo okruhu 800 metrov (čo je cca 10 minút pešej chôdze). Tento prístup značne zredukoval množstvo prehľadávaných zastávok a zrýchlil beh aplikácie.

```
1 def get_nearest_stop(location):
2     stops = {}
3     for name, coord_list in all_stops.items():
4         midpoint = get_midpoint(coord_list)
5         distance_meters = int(vincenty(location, midpoint).meters)
6         # 800 meters is about 10 minutes of walking.
7         if distance_meters < 800:
8             distance = get_distance_meters(location, midpoint)
9             stops.update({name: distance})
10
11     if len(stops) == 0:
12         for name, coord_list in all_stops.items():
13             midpoint = get_midpoint(coord_list)
14             distance = get_distance_meters(location, midpoint)
15             stops.update({name: distance})
16
17     nearest = min(stops.values())
18     for name, distance in stops.items():
19         if distance == nearest:
20             nearest_name = name
21     del stops[nearest_name]
22     return nearest_name
```

Listing 12: Funkcia `get_nearest_stop`

4.5 ImhdsK

ImhdsK je neoficiálna API pre imhd.sk. Táto API posiela GET požiadavku, ktorá obsahuje parametre odkiaľ, kam, dátum a čas, na server a ako odpoveď dostane rovnakú webovú stránku, aká sa zobrazí pri použití imhd.sk v prehliadači. Nasledovne ju automaticky spracuje a nájde v jej štruktúre informácie o spojeniach.

4.6 Backend

V tejto podkapitole ukážeme a popíšeme vytvorené a použité funkcie.

4.6.1 normalize

Funkcia `normalize` odstraňuje špeciálne znaky z názvu zastávky. Funkcia sa ukázala nutnosťou pre úspešné získanie informácií z imhdsK API.

4.6.2 get_coordinates a get_address

Tieto funkcie posielajú požiadavku cez Geopy API. Geopy vracia `Location` objekt, ktorý obsahuje zemepisnú výšku, zemepisnú šírku a adresu. Funkcia `get_coordinates` vráti set súradníc na základe adresy. Funkcia `get_address` vráti adresu na základe súradníc.

4.6.3 get_distance_meters a get_distance_seconds

Funkcia `get_distance_meters` vracia dĺžku pešieho presunu z bodu A od bodu B v metroch. Aby získala vzdialenosť v metroch, vynásobí vzdialenosť v sekundách (od `get_distance_seconds`) priemernou rýchlosťou pešieho presunu človeka, čo je cca 1.7m/sec.

4.6.4 get_midpoint

Funkcia `get_midpoint` nájde a vráti stredný bod z prijatých súradníc.

4.6.5 get_stop_name a get_stop_location

Funkcia `get_stop_name` vráti meno hľadanej zastávky, ak sa nachádza v databáze. Funkcia `get_stop_location` vráti set súradníc hľadanej zastávky, ak sa nachádza v databáze. Rozlišuje medzi autobusovými a električkovými zastávkami.

4.6.6 check_prefered_way

Funkcia `check_prefered_way` vráti preferovaný spôsob prepravy medzi dvoma bodmi. Najprv zistí, či trasa z bodu A do bodu B trvá pešo kratšie ako 15 minút. Ak áno, preferovaný spôsob prepravy bude pešo. Ak by prejsenie trasy trvalo dlhšie ako 15 minút, skontroluje dĺžku trasy spojov. Po porovnaní týchto dvoch časov vráti buď `True`, pre peší presun, alebo `False`, pre presun spojmi.

```
1 def check_prefered_way(frm, to):
2     walk = get_distance_seconds(frm, to)
3     walk = int(walk / 60)
4     if walk <= 15:
5         return True
6
7     bus = 0
8     routes = get_mhd(frm, to)
9     drives = routes[0].drives
10    for drive in drives:
11        bus += int(drive.length.strip(' min'))
12
13    if walk < bus:
14        return True
15    return False
```

Listing 13: Funkcia `check_prefered_way`

4.6.7 identify_line

Funkcia `identify_line` vráti typ spoja, na základe jeho čísla. Vo funkcii je zoznam čísel liniek pre každý typ spoja. Funkcia sa pozrie, či prijaté číslo sa nachádza v aspoň jednom

z troch zoznamov. Predpokladáme, že funkcia vždy prijme existujúce číslo spoja, keďže typ spoja zisťujeme z informácií získaných od imhd.

4.6.8 get_mhd

Funkcia `get_mhd` posiela požiadavku cez `imhds` API. Ku času požiadavky pripočítavame aj číslo `walking_delta`, ktoré zabezpečuje, že výsledky budú rátať aj s časom pešieho presunu na zastávku. Funkcia vráti pole spojov zo zastávky 'frm' na zastávku 'to'.

```
1 def get_mhd(frm, to):
2     if frm == to:
3         print('Location from and to are the same.')
4         return None
5
6     stop_frm = get_nearest_stop(frm)
7     stop_to = get_nearest_stop(to)
8
9     walking_delta = get_distance_seconds(frm, get_stop_location(stop_frm))
10    now = datetime.now()
11    time = now + timedelta(seconds=walking_delta)
12    time = time.strftime("%H:%M")
13
14    routes = imhds.routes(normalize(stop_frm), normalize(stop_to), time=time,
15                           date='')
16
17    if len(routes) == 0:
18        print('No routes found!')
19        return None
20    return routes
```

Listing 14: Funkcia `get_mhd`

4.6.9 find

Funkcia `find` nájde trasy z bodu A do bodu B a vytvorí pole objektov `Route`, ktoré obsahujú informácie o každom potrebnom prestupe a pešom presune.

Pôvodne sme chceli vytvoriť objekt, alebo naplňať dátovú štruktúru, ktorá bude obsahovať všetky potrebné informácie. Neskôr sme sa ale radšej rozhodli využiť objekty, ktoré dostaneme ako odpoveď od imhdsk API a doplniť informácie, ktoré nám chýbajú, teda trasu zo začiatočného bodu na prvú zastávku, trasu z poslednej zastávky do cieľového bodu a doplnkové informácie k existujúcim trasám, ako napríklad súradnice a typ spoja.

Táto funkcia kombinuje všetky predchádzajúce funkcie. Najskôr zistíme, či sa nám oplatí cestovať spojom, alebo pešo. Ak sa nám oplatí cestovať spojmi, funkcia získa pomocou `get_mhd` pole objektov `Route`. Objekty `Route` reprezentujú všetky potrebné presuny - každý `Route` obsahuje pole objektov `Drive`, ktoré reprezentujú jednotlivé presuny, ako napríklad cestu spojom, či peší presun medzi zastávkami, ktoré sú nutné na presun medzi zadanými zastávkami na imhd. Cez toto pole iterujeme a každému `Route` doplníme ďalšie 2 objekty `Drive` - jeden pre trasu z bodu A na zastávku a druhý pre trasu z poslednej zastávky do bodu B, a naplníme ich potrebnými informáciami.

Ak pešo, vytvoríme objekt `Drive`, naplníme ho všetkými potrebnými informáciami, ako začiatočná a koncová lokácia (vrátane súradníc) a dĺžka presunu (v metroch a minútach). Následne sa vytvorí pole, v ktorom sa nachádza len jeden objekt `Route` (s nami vytvoreným objektom `Drive`) a táto funkcia ukončí činnosť.

```

1      ...
2      # First drive from point A to nearest stop (B_1).
3      to_stop = imhdsk.Drive()
4      line_type = identify_line(r[0].drives[0].line)
5      to_stop.start = start
6      to_stop.start_c = list(frm)
7      to_stop.dest = r[0].drives[0].start
8      to_stop.dest_c = list(get_stop_location(to_stop.dest, line_type))
9      to_stop.midpoint = list(get_midpoint([to_stop.start_c,
10                                         to_stop.dest_c]))
11     to_len = get_distance_seconds(frm, to_stop.dest_c)
12     if to_len == 0:
13         # Minimum value for walk from point A to stop.
14         to_len = 1
15     to_stop.length = '{} min'.format(int(to_len / 60))
16     to_stop.dist = int(to_len * 1.7)
17     to_stop.walk = True
18     to_stop.bus = False

```

```
19     to_stop.tbuss = False
20     to_stop.tram = False
21     to_stop.instr = walkinstr.format(to_stop.start, to_stop.dest)
22     ...
```

Listing 15: Funkcia find

Ak konkrétny objekt Drive nereprezentuje peší presun, pridávame informáciu o type spoja - autobus, trolejbus, alebo električka.

Funkcia vracia pole objektov Route, ktoré je pripravené pre čo najjednoduchšiu manipuláciu v našej webovskej aplikácii.

4.7 Frontend

Pre používateľa je dostupná webová stránka s možnosťou zadať odkiaľ a kam ideme. Aplikácia ponúka aj možnosť automatického identifikovania lokácie používateľa. Táto funkcionality je možná vďaka HTML5, ktoré ponúka API pre lokalizačné služby. Používateľ musí súhlasiť s použitím týchto služieb pre nájdenie jeho geografickej pozície. Ak nesúhlasí, je potrebné zadať adresu manuálne.

4.7.1 HTML5 Geolocation

HTML5 Geolocation je API, ktorá umožňuje zistiť geografickú pozíciu používateľa. Nami používaná funkcia `getCurrentPosition()` vracia objekt, ktorý obsahuje zemepisnú šírku a výšku, presnosť nájdennej pozície, nadmorskú výšku (a jej presnosť), smer, rýchlosť a čas merania. Objekt HTML5 Geolocation ponúka aj ďalšie zaujímavé funkcie:

- `watchPosition()` - vráti aktuálnu pozíciu používateľa a následne posiela aktualizovanú pozíciu v pravidelných intervaloch
- `clearWatch()` - zastaví beh `watchPosition()`

Tieto funkcie sú vhodné, ak by sme chceli vyhodnocovať cesty aj počas presunu užívateľa.

4.7.2 Používateľské rozhranie

Pre dizajn stránky sme sa rozhodli použiť rozloženie prvkov, ktoré je dobre čitateľné na osobných počítačoch a zároveň vhodné aj pre mobilné telefóny.

Na hlavnej stránke je zadávanie adresy bodu A a bodu B. Adresu A je možné zadať aj automaticky, použitím tlačítka na lokalizovanie používateľovho zariadenia.

(a) Základná stránka.

(b) Základná stránka s vyplnenými údajmi.

(c) Tlačítko na lokalizovanie zariadenia.

(d) Súhlas so zdieľaním lokácie.

(e) Základná stránka s vyplnenými údajmi - súradnice zariadenia.

Obr. 4.3: Hlavná stránka

Na stránke s výsledkami zobrazujeme inštrukcie, mapy pre pešie presuny a výber z rôznych časových alternatív.

BAKALÁRSKA PRÁCA JAKUB NOVÁK

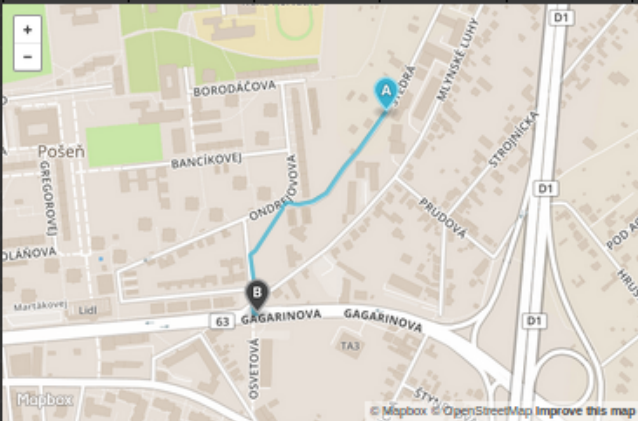
Alternative 1

Alternative 2

Alternative 3

Alternative 4

Alternative 5

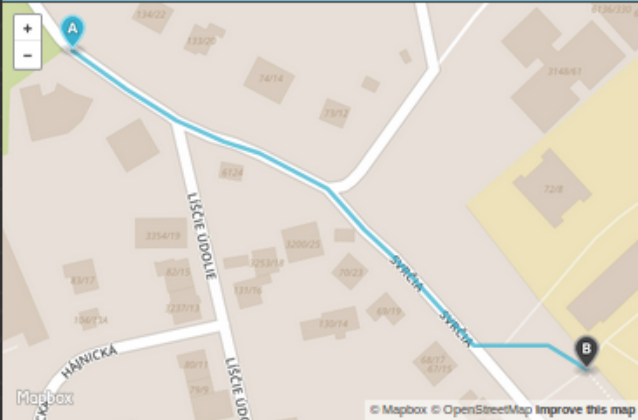


Presunte sa z Štedrá 1 na Ondrejovova.

Odchod: 05:19
Na trolejbusovej zastávke Ondrejovova nastúpte na spoj číslo 202, smer Rajsá.
Vystúpte na zastávke Rajsá.

Odchod: 05:45
Na električkovej zastávke Námestie SNP nastúpte na spoj číslo 9, smer Jurigovo nám..
Vystúpte na zastávke Jurigovo nám..

Odchod: 06:04
Na autobusovej zastávke Jurigovo nám. nastúpte na spoj číslo 139, smer Hájnická.
Vystúpte na zastávke Hájnická.



Presunte sa z Hájnická na Svrčia 6.

Jakub Novák - Copyright 2018

Obr. 4.4: Stránka s výsledkami

Kapitola 5

Rozšírenia

5.1 Zastávky spojov

Zastávky spojov sú momentálne neaktualizované. To znamená, že ak sa zastávka presunie, alebo zruší, nevieme to inak, ako manuálne napraviť.

Do budúcnosti by sme radi našli spôsob, ako zastávky udržiavať aktualizované, prípadne k nim pridali ďalšie údaje, ako smer (na ktorej strane cesty sa nachádzajú) a zoznam spojov, ktoré na danej zastávke stoja, primárne pre zastávky s rovnakým názvom.

V súčasnosti absenciu týchto informácií riešime nájdením stredného bodu zastávok s rovnakým názvom, čo samozrejme nie je optimálne. Pre väčšinu prípadov to postačuje, avšak vyžaduje to, aby sa používateľ podľa ostatných zobrazených informácií (číslo spoja, smer) presunul na správnu zastávku v okolí stredného bodu bez navigácie.

5.2 Používateľská odozva

Používateľská odozva je dôležitá časť tohto projektu. Používatelia by mali možnosť ohodnotiť ich skúsenosť s vybranou cestou. Meškal niektorý zo spojov? Bol spoj preplnený? Tieto a ďalšie pripomienky by boli zodpovedané vždy po úspešnom príchode do bodu B. Prispievali by tak ku prioritizácii spojov. Odozva by samozrejme obsahovala aj čas, aby sme čo najpresnejšie mohli navigovať cestujúcich v dopravnej špičke, aj mimo nej.

Úzko spojená so spätnou odozvou používateľov je prioritizácia spojov. Každý spoj by obsahoval dátovú štruktúru, ktorá by pre každý úsek udržiavala jeho prioritu na základe štatistiky meškaní. Po získaní spojov (teda zoznamu objektov Route), by sa skontrolovali úseky, cez ktoré by sme chceli navigovať používateľa. Ak by mali všetky úseky rovnakú prioritu, poslali by sme používateľa prvým nájdeným výsledkom. Ak by nejaký spoj mal vyššiu prioritu a celkový čas trasy by trval menej, než čas prvej nájdennej trasy, vybrali by sme trasu s týmto spojom.

5.3 Výluky

Zakomponovanie výluk spojov by výrazne zefektívnilo vyhľadávanie trás. Aplikácia by si pamätala časový rozsah a ktoré spoje majú výluky a automaticky by vyhľadávala trasy, ktoré tieto spoje neobsahujú.

5.4 Sledovanie zariadenia navigovaného používateľa

Ďalším možným rozšírením je využitie sledovania polohy používateľa na zlepšenie budúcej navigácie. Vďaka týmto informáciám by sme mohli nahradiť priemernú rýchlosť pešieho presunu človeka a každému používateľovi priradiť jeho vlastnú rýchlosť pohybu, čo by viedlo k lepším výsledkom vyhľadávania.

Sledovanie zariadenia by taktiež prispelo k zautomatizovaniu používateľskej odozvy. Aplikácia by si mohla porovnať, či cesta skutočne trvala toľko, koľko mala. Ak nie, vedela by presne identifikovať úsek, podľa času, na ktorom nastala odchýlka od predpokladaného príchodu a upravila by jeho prioritu. Týmto by sa zrušila akákoľvek potreba pre nutnú odozvu od používateľa a zostala by dobrovoľná, s doplnkovými informáciami o kvalite trasy.

Ak by sme chceli dynamickejšiu navigáciu, mohli by sme využiť, informácie o pozícii na pravidelné aktualizovanie trasy. Ak by iný používateľ odoslal odozvu, že spoj mešká, vedeli by sme presmerovať ostatných používateľov na iný spoj, ktorý by meškanie nemal.

5.5 Rozšírenie o ďalšie mestá

Samozrejme, našu aplikáciu by sme radi eventuálne rozšírili o podporu aj pre ďalšie mestá. Ideálne by bolo, najprv migrovať našu jednoduchú databázu do SQL databázy. To by nám umožnilo pridať možnosť jednoducho prepínať medzi databázami rôznych miest.

5.6 Chat a zdieľanie

Interaktívny chat by mohol byť využitý dvoma spôsobmi.

Používatelia by mohli spolu zdieľať svoju lokáciu a navzájom sa sledovať na interaktívnej mape. Skupinový chat by bol samozrejme prístupný, aby nebolo potrebné prepínať medzi našou a inou aplikáciou.

Naša aplikácia už ponúka možnosť výberu z 5 alternatív. Stalo sa v premávke niečo nečakané a nepredvídane? Používatelia by mohli o tom informovať ostatných, aby mali šancu vyhnúť sa spomaleniu výberom inej alternatívy.

5.7 Pomoc nevidiacim

Mnoho aplikácií, ktoré sa snažia navigovať po meste nemyslí na menšinu nevidiacich a slabozrakých. Existujú projekty, napríklad v Poľsku [15] a Fínsku [16], ktoré tento sa tento problém snažia riešiť.

Aplikácia by mohla mať mód pre nevidiacich. V tomto móde by boli inštrukcie podávané používateľovi zvukom, alebo vibráciami - napríklad 1 vibrácia pre posun rovno, 2 vibrácie pre otočenie vľavo, 3 vibrácie pre otočenie vpravo, atď. Pre funkčnosť takéhoto systému by boli potrebné presné informácie o meškaní a príchodoch spojov.

Kapitola 6

Testy

Rýchlosť vyhľadania spoja vzhľadom na celkovú vzdialenosť, vzdialenosť od najbližšej zastávky a počet prestupov. Cieľom týchto testov bolo zistiť, či rýchlosť našej aplikácie závisí od niektorého z vyššie uvedených parametrov.

z	do	k prvej zastávke [m]	celková vzdialenosť [m]	počet prestu- pov	rýchlosť vyhľadania [ms]
Cesta mládeže	Běžová	429	17909	3	4661.2
Ihličnatá	Botanická	768	15298	2	5798.8
Chotárna	Jasovská	230	4992	1	6484.6
Hodonínska	Pestovateľská	336	17395	4	6561.6
Šaštínska	Micherov majer	10	15016	4	6993.4
Popradská	Botanická	455	13500	2	7276.6
Štedrá	Svrčia	402	11370	3-4	8475
Rovná	Kollárovo nám.	238	6209	2	15579.2

Tabuľka 6.1: Rýchlosť vyhľadania spoja

Výsledná rýchlosť vyhľadania v tabuľke je priemer rýchlostí všetkých testovaných behov funkcie. Rýchlosť behu funkcie sa ukázala byť nezávislá od celkovej vzdialenosti, vzdialenosti bodu A od prvej zastávky, aj počtu spojov potrebných na presun.

Kapitola 7

Inštalácia a spustenie

V tejto kapitole popíšeme, ako nainštalovať potrebné knižnice a spustiť server pre fungovanie našej webovej aplikácie.

Virtuálne prostredie

Aby sme náhodou neporušili hlavnú inštaláciu pythonu, použijeme pre našu aplikáciu virtuálne prostredie, v ktorom sa bude všetko inštalovať. Vďaka tomuto virtuálnemu prostrediu nie je nutné vlastniť administrátorské práva pre všetky ďalšie potrebné inštalácie.

Ak používate python verziu staršiu ako 3.4, bude potrebné stiahnuť a nainštalovať virtuálne prostredie týmto príkazom:

```
sudo apt-get install python-virtualenv
```

Ak používate python verziu 3.4, alebo mladšiu, presuňte sa v termináli do priečinka s našou aplikáciou a zadajte príkaz:

```
. venv/bin/activate
```

Flask

Ďalej je potrebné nainštalovať flask. To dosiahneme zadáním nasledovného príkazu do terminálu:

```
pip3 install Flask
```

ImhdsK

Pre inštaláciu imhdsK, zadajte:

```
pip3 install imhdsK
```

Mapbox

Pre inštaláciu mapbox, zadajte:

```
pip3 install mapbox
```

V súbore bcf.py je možné zmeniť mapbox token, na token svojho Mapbox účtu zmenou hodnoty premennej MAPBOX_TOKEN.

Geopy

Pre inštaláciu geopy, zadajte:

```
pip3 install geopy
```

Spustenie aplikácie

Ak virtuálne prostredie nie je zapnuté, zapneme ho príkazom:

```
. venv/bin/activate
```

Následne zadáme do terminálu príkaz:

```
python3 start.py
```

Ostáva nám len v prehliadači otvoriť stránku <http://localhost:5000/>.

Záver

Úspešne sa nám podarilo vytvoriť webovú aplikáciu na vyhľadávanie najrýchlejšej trasy z bodu A do bodu B pomocou mestskej hromadnej dopravy v Bratislave, napísanú v jazyku Python. Na vytvorenie aplikácie sme použili rôzne knižnice, ktoré nám dopomohli k nášmu cieľu. K našej aplikácii sme vytvorili dizajn, ktorého cieľom je dobrá čitateľnosť aj v mobilnom prehliadači. Táto aplikácia je zverejnená ako open-source projekt na stránke <https://github.com/JakubNvk/bc>. Pri vývoji aplikácie sme sa držali konvencie PEP-8 programovania v Pythone a konvenciám v komunite Flask frameworku.

Najväčší problém, s ktorým sme sa stretli, bolo získavanie údajov o zastávkach a spojoch. Mnohé zo získaných zastávok mali rovnaké názvy a nevedeli sme ich správne priradiť ku spojom, čo sme vyriešili vytvorením tzv. stredného bodu, ktorý slúži ako všeobecný bod pre zastávku s daným názvom a taktiež aj ako bod, okolo ktorého je mapa neskôr vycentrovaná. Vidíme, že na rozšírenie a zefektívnenie aplikácie by sme potrebovali spoluprácu s Dopravným podnikom, aby sme sa zbavili potreby používať knižnicu imhdk, na zistenie informácií o zastávkach a spojoch. Rozšírenie na ďalšie mestá je možné vďaka modularite.

V budúcnosti by sme chceli doimplementovať primárne používateľskú odozvu na základe ktorej by sme zefektívnili vyhľadávanie trás. Ďalej by sme chceli skvalitniť informácie o zastávkach, pridať sledovanie zariadenia používateľa a rozšíriť aplikáciu na ďalšie mestá. Eventuálne by sme radi pridali aj poradcu, ktorý by sledoval aktuálnu situáciu v doprave a odporúčal, napríklad, prestúpiť na iný spoj, ak sa nachádzate v zápche.

Našu aplikáciu sme podrobili testom, ktorých cieľom bolo zistiť, či počet prestupov, vzdialenosť najbližšej zastávky, alebo celková vzdialenosť ovplyvňuje rýchlosť aplikácie. Zistili sme, že žiaden z týchto parametrov rýchlosť neovplyvňuje. Priemerná rýchlosť vyhľadania je 7-8 sekúnd, čo je dostatočne rýchly beh aplikácie pre naše potreby.

Na našej aplikácii sa snažíme stále pracovať.

Literatúra

- [1] OpenStreetMap. Openstreetmap features, Retrieved February 5, 2016. http://wiki.openstreetmap.org/wiki/Map_Features.
- [2] OpenStreetMap. Openstreetmap routing, Retrieved February 5, 2016. <http://wiki.openstreetmap.org/wiki/Routing>.
- [3] Ojw. Pyroute, Retrieved February 6, 2016. <http://wiki.openstreetmap.org/wiki/Pyroute>.
- [4] Ojw. Simpleosmrouter, Retrieved February 6, 2016. <https://github.com/F6F/SimpleOsmRouter>.
- [5] OpenStreetMap. Overpass api, Retrieved February 6, 2016. http://wiki.openstreetmap.org/wiki/Overpass_API.
- [6] Wikipedia. Google maps, Retrieved February 5, 2016. https://en.wikipedia.org/wiki/Google_Maps.
- [7] Google. Google directions, Retrieved February 6, 2016. <https://developers.google.com/maps/documentation/directions/intro>.
- [8] Google. Google geocoding, Retrieved February 6, 2016. <https://developers.google.com/maps/documentation/geocoding/intro>.
- [9] Google. Google maps api tos, Retrieved February 7, 2016. <https://developers.google.com/maps/terms?csw=1#6-google-proprietary-rights>.
- [10] Google. Google pricing and plans, Retrieved February 6, 2016. <https://developers.google.com/maps/pricing-and-plans/#details>.
- [11] Wikipedia. Bing maps platform, Retrieved February 5, 2016. https://en.wikipedia.org/wiki/Bing_Maps_Platform.
- [12] Microsoft. Bing route, Retrieved February 6, 2016. <https://msdn.microsoft.com/en-us/library/mt270292.aspx>.

- [13] Microsoft. Bing location, Retrieved February 6, 2016. <https://msdn.microsoft.com/en-us/library/ff701725.aspx>.
- [14] Microsoft. Bing maps api tos, Retrieved February 7, 2016. <https://www.microsoft.com/maps/product/terms.html>.
- [15] Michał Markiewicz and Marek Skomorowski. Public transport information system for visually impaired and blind people. In *Transport Systems Telematics*, pages 271–277. Springer, 2010.
- [16] Koskinen Sami and Virtanen Ari. Public transport real time information in personal navigation systems for special user groups. In *11th ITS World congress. Nagoya, Japan*, page 6, 2004.
- [17] OpenStreetMap. Openstreetmap comparison, Retrieved February 5, 2016. http://wiki.openstreetmap.org/wiki/Comparision_Google_services_-_OSM.

Dodatok A

Elektronická verzia práce na CD

Priložený disk CD obsahuje zdrojové kódy práce, ktoré sú za pomoci kapitoly 7 spustiteľné pod systémom Linux.