Diplomová práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Plánování trasy s pomocí otevřených dat

Bc. Martin Kostohryz

Vedoucí: RNDr. Ondřej Žára
Obor: Kybernetika a robotika
Studijní program: Kybernetika a robotika
Květen 2021

Poděkování

Děkuji svému vedoucímu, že mi umožnil na této práci pracovat a všem, kdo mi pomohli aplikaci otestovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 21. května 2021

Abstrakt

Účelem této práce je vytvořit aplikaci, která by dokázala plánovat cestu různými dopravními prostředky po Praze. Kromě plánování aplikace zobrazuje mapu, do které lze dynamicky přidávat a odebírat prvky dopravní infrastruktury. Důraz je kladen na zobrazení reálné polohy sdílených dopravních prostředků od různých dodavatelů. Data zobrazená v aplikaci vychází z volně dostupných opendat, které poskytuje Hlavní město Praha na svých portálech.

Klíčová slova: plánování trasy, městská doprava, mapy, API, otevřená data

Vedoucí: RNDr. Ondřej Žára

Abstract

This thesis aims to create an app that could plan a route in Prague by various kinds of transport. Moreover, the app displays a map that contains dynamically added and removed elements of transport infrastructure. Emphasis is placed on displaying the real location of shared vehicles from various suppliers. The data displayed in the application is based on the freely available open data provided by Prague on its portals.

Keywords: route planning, urban transport, maps, API, open data

Title translation: Route planning using open data

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Doprava ve velkých městech je čím dál větší problém. Mnoho lidí, kteří se každý den po městě pohybují řeší jak se na dané místo dostat a to s ohledem na mnoho kritérií. Nejde jen o způsob jak se do daného místa dostat nejrychleji, ale i pohodlí, cenu a další faktory cesty. Druhů dopravy ve městě přibývá, ať už jde o nové sdílené auta a kola, ale i rozmach nových taxislužeb, které díky své nízké ceně přilákali nové zákazníky.

Každý druh dopravy má své pozitiva i negativa a podle nich i své příznivce a odpůrce. V posledních pár letech se staly fenoménem sdílené dopravní prostředky. Jsou podporované městy a získávají si čím dál větší oblibu i u lidí. Dříve si lidé půjčovali jen auta a kole pro speciální příležitosti. Postupem času se sdíleným prostředků začalo dařit a v ulicích přibyly další kola a auta, ale i koloběžky a skůtry.

Tato práce je zaměřena na tvorbu aplikace, která bude umožňovat naplánovat trasu po Praze. Webová aplikace taktéž ukáže na mapě prvky, které souvisí s dopravou po městě. Jednou z klíčových vlastností bude zobrazení sdílených prostředků v mapě. Uživatel si tak bude moct naplánovat trasu autem nebo na kole, i když kolo ani auto nemá a musí si ho nejdříve půjčit.

Celá aplikace bude vycházet z otevřených dat, která jsou volně dostupná na internetu. Využita budou především opendata Prahy. Funkčnost aplikace bude na těchto datech závislá. Aplikace tak ukáže kolik užitečných dat o sobě město veřejně poskytuje a jak se dají využít.

Část I

Teoretická část

Kapitola 2

Otevřená data

2.1 Koncept otevřených dat

Otevřená data jsou jakákoliv data zveřejněna na internetu bez omezení na použití. Vydavatel k datům poskytuje právní svolení k jejich použití. Data by měla být snadno dostupná a ve strojově čitelném formátu. Využití by nemělo být nijak podmíněno [dvpp].

Zveřejňování dat přináší mnoho výhod. Zprvu byla data brána především jako nástroj transparentnosti. Při zveřejnění informací o chodu státní správy se snáze odhalí všemožné podvody a nesrovnalosti v řízení státu. Následně se ukázalo, že data mohou být i ekonomicky využitelná. Mohou vznikat nové služby založené na opendatech, ze kterých benefitují občané. Státu, potažmo obcím se vyplatí data tedy zveřejňovat, protože tím dávají prostor k inovacím a tvorbě nových služeb na svém území[Gro21]

2.2 Opendata v ČR

Otevřená data jsou v českém prostředí zakotvená v zákonu o svobodném přístupu k informacím (106/1999 Sb.) [cit99]. Ten přímo definuje opendata jako "informace zveřejňované způsobem umožňujícím dálkový přístup v otevřeném a strojově čitelném formátu, jejichž způsob ani účel následného využití

není omezen a které jsou evidovány v národním katalogu otevřených dat“. Národním katalog otevřených dat je umístěn stránkách `data.gov.cz`. Na stránkách jsou k nalezení návody pro poskytovatele na celý proces zveřejnění dat. Součástí je katalog, který ke květnu 2021 obsahuje přes 135 000 datových sad.

2.3 Opendata v Praze

Praha začala poskytovat otevřená data v roce 2002 kdy IPR Praha (Institut plánování a rozvoje) začal sdílet geografické data Prahy[IparhmP]. K dalšímu rozvoji došlo v roce 2012 kdy byl spuštěn pražský geoportál [htt21a]. Od té doby došlo k velkému rozvoji a v současné době Praha zveřejňuje data ze všech oblastí své působnosti. Datové sady zpřístupňují jak jednotlivé radnice a magistrát tak i městem vlastněné společnosti. Kompletní přehled datových sad je k nalezení na opendata.praha.eu. Pro účely této práce jsou zajímavé především dva portály s opendaty a to Golemio a Geoportal.

2.3.1 Geoportál

Na stránkách Geoportálu jsou k nalezení online mapy, aplikace a hlavně geografická data [htt21a]. Jsou tam data o všech ulicích a cyklostezkách, které budou využity k plánování trasy. Vytvářená aplikace dále využije sady dat týkající se parkování a veřejné dopravy.

2.3.2 Golemio

Platforma Golemio je na své webové stránce definována jako: soubor technických nástrojů pro integraci, ukládání, vizualizaci a poskytování dat, a zároveň je to tým odborníků na městská data. Cílem datové platformy Golemio je poskytovat kvalitní IT služby magistrátu a městským částem v oblasti zpracování Smart City dat."[OI18]. Pro účely této práce je nejdůležitější poskytování dat. Platforma poskytuje data formou REST API. Dokumentace je dostupná na stránkách golemioapi.docs.apiary.io. Z poskytovaných dat budou využity data o sdílených kolech a autech. V době psaní této práce jsou k dispozici data od pěti poskytovatelů sdílených aut a jednoho poskytovatele sdílených kol.

Kapitola 3

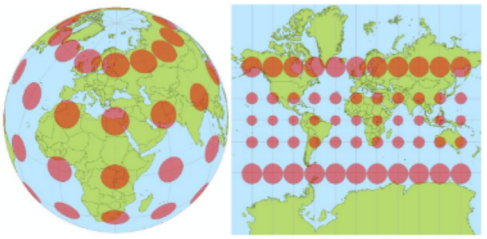
Geografická data a jejich formáty

3.1 Mapové zobrazení

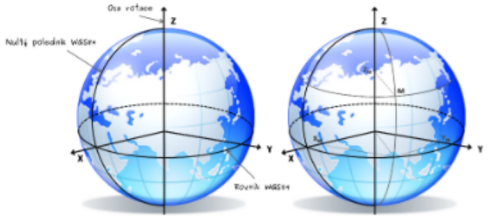
Pro zobrazení mapy je zapotřebí data a mapové podklady reprezentovat ve stejných souřadnicích a za použití stejného zobrazení. Existuje mnoho různých zobrazení, přičemž každé z nich má své výhody a nevýhody. Zobrazení se liší především svojí přesností.

Druhy zobrazení:

- **Mercatorovo zobrazení**
  Toto zobrazení je nejrozšířenější a je využíváno většinou mapových aplikací. Jde o zobrazení, které vznikne vložením země do válce tak aby se válce země dotýkala na rovnících a kolmého promítnutí povrchu země na válec. Válec je následně rozbalen, čímž vzniká mapa. Zobrazení je přesné na rovníku a směre k obou pólům se přesnost zmenšuje [GIS14]. Obrázek níže zobrazuje k jakému zkreslení dochází.
- **S-JTSK**
  Systém Jednotné Trigonometrické Sítě Katastrální, který je nejčastěji používán v ČR. Jde o tzv. Křovákovo zobrazení, které navrhl Ing. Josef Křovák roce 1922.[VK]
- **WGS84**
  Geodetický standard vydaný ministerstvem obrany USA v roce 1984. Jedná se o pravotočivý systém pevně spojený se zemí [cit].



Obrázek 3.1: Demonstrace zkreslení na kontrolních kruzích (Zdroj: Stefan Kühn)



Obrázek 3.2: Referenční systém a pravoúhlé souřadnice (zdroj: Royal Observatory of Belgium)

3.2 Formáty dat

Ať už je využit kterýkoliv ze systémů, data musí být uložena v paměti v nějakém formátu. V průběhu let bylo vytvořeno mnoho formátů pro uložení geografických dat. Například data na pražském geoportálu, který je využíván jako zdroj dat k této práci, jsou uložena v těchto formátech: [**?**]

- **shapefile**
- **geoJSON**
- **GML**
- **DXF**
- **Rastrových formátech**
  - **TIFF**
  - **JPG**

Kapitola 4

Vyhledávání trasy

Problém vyhledávání trasy z místa A do místa B spočívá v nalezení cesty, která bude podle určitého kritérii nejlepší. Může se jednat o nejkratší cestu z hlediska vzdálenosti, času nebo například celkové ceny cesty. Cesta se plánuje podle dopravního prostředku v síti cest daného typu. Příkladem takové sítě může být silniční síť. Dopravní síť může být reprezentována jako obecný graf, kde vrcholy reprezentují křížení cest a hrany cesty mezi nimi.

4.1 Algoritmy pro vyhledání nejkratší cesty

Pokud je možné prostředí, ve kterém se hledá nejkratší cesta převést na graf je možné použít následující algoritmy:

4.1.1 Dijkstrův algoritmus

Algoritmus byl poprvé publikován v roce .1959 E. W. Dijkstrou [TA]jde o algoritmus, který začne prohledávat sousední vrcholy počátečního vrcholu a uloží si je do seznamu všech navštívených vrcholů. Ze seznamu následně vybírá vždy ten nejbližší neprohledaný vrchol a přepočítá u něj vzdálenost od počátečního vrcholu. Následně prohledá všechny jeho sousedy a přidá je do seznamu. jakmile je vybrán ze seznamu koncový vrchol prohledávání

končí. V případě že u každého uzlu si algoritmus uloží i uzel, ze kterého se do něj dostal, je možné zpětným průchodem seznamu od koncového uzlu k počátečnímu uzlu nalézt nejkratší cestu .

Nevýhodou tohoto prohledávání je, že algoritmus prohledává graf na všechny strany stejně. To v případě že známe pozici a tudíž i směr ke koncovému uzlu není příliš efektivní. Je velmi pravděpodobné, že nejkratší cesta se bude nacházet ve směru ke koncovému uzlu.

4.1.2 Algoritmus A-star

Problém s Dijkstrovím algoritmem řeší algoritmus A-star. Ten vychází z předchozího algoritmu. liší se v tom, že přidává k ohodnocení uzlů navíc heuristiku. Ta muže být v případě geografických dat reprezentována vzdušnou vzdáleností mezi daným uzlem a koncovým bodem [Roy19].

4.1.3 Obousměrné prohledávání

Další možnost jak vyhledat trasu je použít obousměrné prohledávání. Graf se začne prohledávat z obou stran najednou a ve chvíli, kdy je některý uzel navštíven a prohledán z obou stran dojde k ukončení algoritmu. Výsledná cesta je spojená z cest z obou stran k vrcholu kde došlo k propojení.

Výhodou tohohle přístupu je, že šetří paměť. Při prohledávání grafu kvadraticky narůstá spotřeba paměti a času. U obousměrného prohledávání je tak ušetřen výpočetní čas i paměť

4.2 Zhodnocení algoritmů

Při prohledávání většího grafu mohou mít zmíněné algoritmy problém, protože jsou příliš časově náročné. U vyhledávání trasy na velké vzdálenosti se tedy využívají pokročilejší algoritmy, které využívají například předzpracování dat. Webová aplikace tvořená v této práci má vyhledávat trasy po Praze, tudíž graf nebude natolik rozsáhlí aby nebylo možno ho prohledat dříve zmíněnými algoritmy.

Část II

Praktická část

Kapitola 5

Návrh aplikace

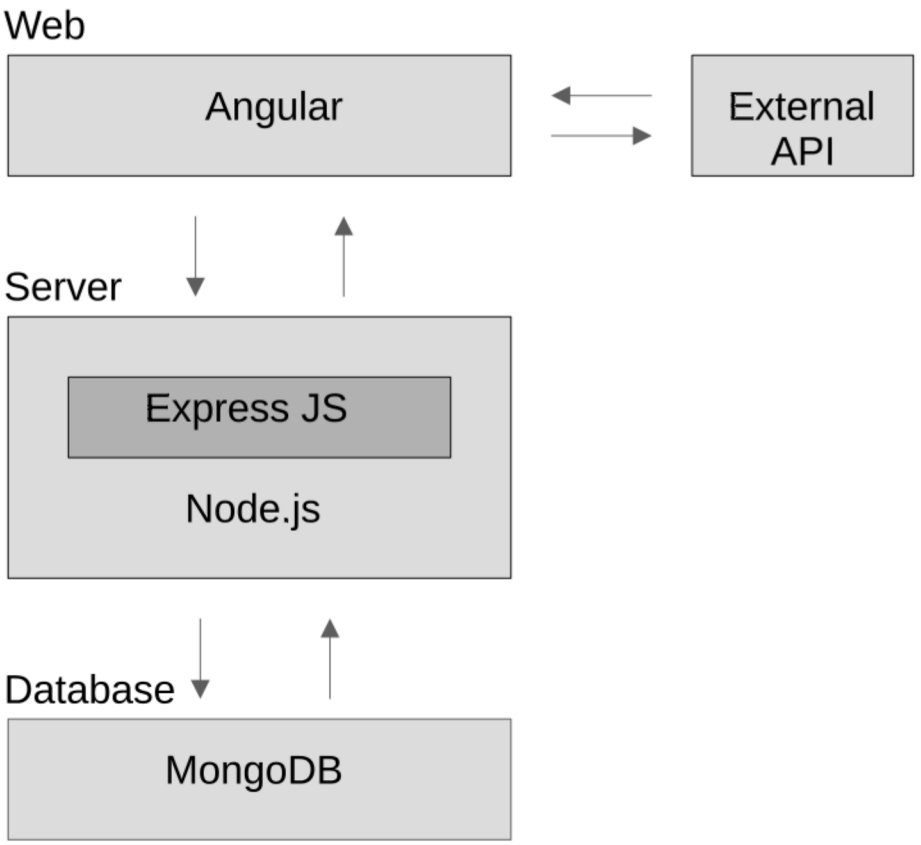
V této kapitole bude popsána architektura celé aplikace a její jednotlivé části.

5.1 Požadavky na aplikaci

Aplikace si dává za cíl umožnit lidem vyhledat trasu po Praze různými dopravními prostředky. Uživatel bude moct zadat počáteční a koncový bod trasy na mapě. Bude na výběr více druhů podkladových map, mezi kterými bude možné přepínat. Aplikace nabídne 3 módy. Kolo, Auto a městská doprava. V každém módu půjde filtrovat relevantní prvky pro daný mód. Aplikaci bude dále obsahovat lokalizaci uživatele, ze které taktéž bude moc být naplánována cesta.

5.2 Architektura aplikace

Aplikace je koncipována jako MEAN stack. To odpovídá použitým technologiím Mongodb, Express, Angular, a NodeJS. Schéma jak jsou technologie propojené je vidět na obrázku níže. Klasický MEAN stack je rozšířen ještě o komunikaci frontedu s golemio API odkud jsou brána některá data.



Obrázek 5.1: Architektura aplikace

Kapitola 6

Data a databáze

Jako první bude popsána databáze a s tím i související nahrání dat. Jako databáze bude použito MongoDb [Mon]. Tato databáze ukládá data ve formátu JSON-like, což je výhodné jelikož tento formát bude použit i na frontendu. Data jsou ukládaná do dokumentů, které jsou ukládány v kolekcích.

6.1 Data

Jak již bylo zmíněno aplikace vychází z opendat. V databázi bude potřeba uchovat geodata obsahující informace o parkovacích místech a parkování obecně, zastávkách MHD vybavení pro kola, cyklostezkách a silnicích. Všechny tyto data je možné získat na stránkách Geoportálu prahy. Data jsou dostupná v různých formátech. Pro tuto aplikace bylo zvolen formát GeoJson pro svou jednoduchost. Samostatnou částí bude vytvořit orientované grafy pro vyhledávání trasy, ty budou popsané v části zabývající se implementací algoritmu na vyhledání trasy. V následující tabulce je vidět struktura databáze.

6.2 Spuštění databáze

MongoDb je možné spustit a provozovat více způsoby. Databázi je možné nainstalovat přímo na server nebo je možnost využít cloudového řešení Atlas

Název kolekce	Detail
cycle_path	Každý dokument obsahuje informace o jedné cyklostezce. Obsaženy jsou innformace obsahují jak pozici cyklostezky tak její detailní informace.
cycle_graph_nodes	Obsahuje vrcholy grafu cyklotras, který se využívá při vyhledávání trasy.
parkingLot	Kolekce velkých parkovišť provozovaných Prahou.
paidParking	Kolekce oblastí ulic kde je zpoplatněno parkování včetně ceny.
ZTP	Místa parkování pro zdravotně postižený.
prohibitedParking	Oblasti ulic kde je parkování zakázáno.
streets	Kolekce všech komunikací včetně jejich zatřídění.
street_graph_nodes	Obsahuje vrcholy grafu komunikací, který se využívá při vyhledávání trasy.
PID_lines	Obsahuje data o jednotlivých linkách MHD.

Tabulka 6.1: Struktura databáze

[Mon21]. Jelikož i zbytek aplikace bude následně nahráno do cloudu, Byl vybrán druhý způsob. Atlas navíc velmi zjednodušuje celý proces zprovoznění databáze.

Pro spuštění databáze je nutné se zaregistrovat. V průběhu registrace dochází i k základnímu nastavení databáze. Atlas jako takový musí běžet na nějakém cloudu. Na výběr jsou 3 možnosti Google cloud[httb], AWS [AWS21], Azure [htta]. Následně se vytvoří projekt, který celou databázi reprezentuje. V projektu se vytvoří cluster, ve kterém budou data uložena. Pro účely této práce stačí vytvořit nejmenší možný. Po doladění pár dalších nastavení je cluster vytvořen. Dále je zapotřebí nakonfigurovat připojení tím že se přidají důvěrné IP adresy, ze kterých bude cluster přístupný. Poslední krok v nastavení je vytvoření uživatelů a nastavení jejich práv.

6.3 Nahrání dat

K nahrání dat do databáze je možné využít aplikaci compass, která umožňuje propojení k databázi a grafické znázornění dat [dvpp]. Data stažená z Geoportálu prahy jsou strukturovaná jako jeden GeoJson. Nejprve jsou tedy rozdělena na jednotlivé prvky a pak nahrány do databáze do příslušné kolekce. Každý dokument by měl obsahovat informace o jednom samostatném prvku v mapě (např stojanu na kolo nebo parkovišti).

6.4 Nastavení kolekcí

Aplikace bude vyhledávat dokumenty nejčastěji podle jejich polohy. Je tedy nutné umožnit vyhledávání pomocí vzdálenosti od určitého bodu. Toho lze docílit nastavení 2dsphere indexu v nastavení každé kolekce. MongoDb tak dokáže počítat vzdálenosti dokumentu od sebe vyhledat například všechny dokumenty, které mají souřadnice v určitém okruhu.

Kapitola 7

Backend

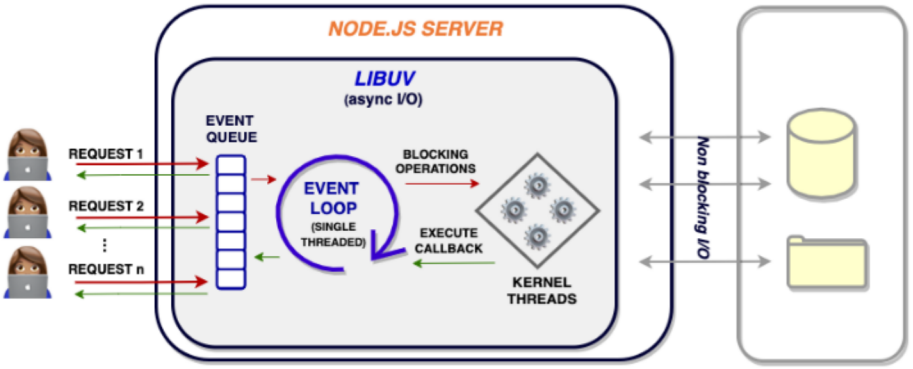
V této části textu bude popsána další část aplikace, kterou je RESTful API server. Server zprostředkovává komunikaci mezi databází a klientskou částí aplikace. Taktéž obstarává vyhledávání trasy.

7.1 Použité technologie

7.1.1 NodeJs

Server je vytvořen za pomocí prostředí NodeJS. Jde o prostředí, které umožňuje pouštět kód v JavaScriptu mimo webový prohlížeč. Hlavní výhoda spočívá v možnosti použít stejný programovací jazyk jak na serveru tak i v klientské části aplikace. NodeJs je založen na Chrome V8 JavaScript enginu, který je používán v prohlížeči Chrome. Kromě běžných funkcí využívaných v prohlížečích je prostředí rozšířeno o možnost využívat systémové zdroje.

Velkou výhodou NodeJs je řešení jeho architektury. Základem je smyčka událostí. Do smyčky vstupují všechny události a jsou jim přiděleny nezávislá vlákna. Díky nezávislosti vláken jsou požadavky vyřizovány paralelně a nedochází k blokování událostí. Tento systém je pak velmi výhodný pokud na server přichází velké množství dotazů.



Obrázek 7.1: Architektura NodeJS
(převzato z: https://cdn.buttercms.com
/0Nh1yR6SSPwqnsKYSfHa)

7.1.2 Express

V rámci NodeJs je dále využit framework Express.js. Jde o open source framework pod licencí MIT, který obsahuje mnoho funkcí užitečných při vytváření mobilních a webových aplikacích [Soec21]. Jednou z užitečných funkcí je například přidání middlewaru při obstarávání http dotazu.

7.2 Implementace

Tato část se bude zabývat konkrétní implementací serveru.

7.2.1 Struktura aplikace

Aplikace má následující strukturu. Hlavní částí je soubor `app.ts`, který slouží jako vstupní bod aplikace. Dále ve složce `Routes` jsou definované třídy pro obsluhu dotazů na server. Poslední část pak obstarává komunikaci s databází.

```
app.ts
/routes
    /common
        common.routes.config.ts
    /parking
        parking.routes.config.ts
        parking.service.ts
    /graph...
    ...
/database
    mongo.service.ts
```

Obrázek 7.2: Struktura kódu

7.2.2 App.ts

jak již bylo zmíněno vstupním bodem aplikace je soubor `app.ts`. V tomto souboru je vytvořena Express aplikace a nadefinovaný http server. Nadefinovány jsou i další pomocné knihovny

7.2.3 Routes

Ve složce jsou definované třídy pro obsluhu dotazů na server. Nejprve ve složce `common` je abstraktní třída `CommonRoutesConfig`, od této třídy dědí všechny ostatní konfigurační třídy pro jednotlivé zdroje.

Ve složkách pro jednotlivé zdroje jsou definovány vždy 3 třídy. `RoutesConfig`, `service` a `Controller`

- `RoutesConfig` je třída, které dědí od zmíněné `CommonRoutesConfig`. V konstruktoru se nadefinuje jméno zdroje a v metodě `configureRoutes` je nadefinována obsluha http dotazů. Jelikož všechna data uložená na serveru jsou určená pouze pro čtení, je definovaná pouze http metoda `get()`.
- Dalším článkem je kontroler, který je volán z předchozí třídy pokud je dotaz na server validní. Kontroler má za úkol dotaz zpracovat a dotázat se pomocí služby na konkrétní data z databáze.
- `service`, nebo česky služba, složí jako poslední stupeň, který má za úkol komunikaci se službou, která už přímo komunikuje s databází.

Celý kód je takto rozdělen do více stupňů tak aby byly od sebe oddělené části aplikace, které spolu nesouvisí a byl dodržen single-responsibility princip.

7.2.4 Komunikace s databází

Poslední část, která zajišťuje komunikaci s databází je služba `mongo.service.ts`. V této službě je definováno připojení k databázi. Služba definuje následující metody pro získání dat:

- `getById` Vrátí jeden dokumet s příslušným id.
- `getOne` Vrátí první dokument, který splňuje filtr.
- `getMany` Vrátí všechny dokumenty z dané kolekce, které vyhovují filtru.
- `runMongo` Metoda je volána hned po startu pro obstarání připojení k databázi.

7.3 Vyhledání trasy

Nejdůležitější částí aplikace je algoritmus pro vyhledávání trasy. Jeho implementace je schovaná ve složce `graph`. Třídy `graph.routes.config.ts` a `graph.service.ts` jsou podobné jako u ostatních zdrojů. Algoritmus je implementovaný v souboru `graph.controller.ts`

7.3.1 Graf

Algoritmus A-star vyhledává v grafu a tak tento graf musí být nejprve vytvořen z dostupných dat. Vytvořeny jsou celkem 3 grafy podle typu dopravního prostředku. Graf vytvořený z ulic, chodníků a cyklostezek.

- Graf z ulic Graf vychází z dat na geoportalu praha, konkrétně jde o "Uliční úseky TSK včetně zatřídění komunikací". Data byla stažena ve formátu GeoJSON v geodetickém standardu WGS 84. Každý dokument

obsahuje údaje části silnice mezi dvěmi křižovatkami. Tudíž dokumenty představují všechny hrany v tvořeném grafu.

- Graf z cyklostezek Cyklostezky ve stejném formátu jako ulice jsou na geoportálu pod názvem "Cyklistické trasy". Postup vytvoření grafu je stejný jako v případě ulic.
- Graf z pěších tras Situace s pěšími trasami je o něco složitější. Dostupné jsou na geoportálu ve formátu DXF, což je CAD formát. Data tedy musí být ze souboru přečtena a uložena jako GeoJSON a následně upravena stejně jako předchozí dva grafy

7.3.2 Implementace algoritmu

Algoritmus na vstupu dostane souřadnice 2 bodů (počátečního a koncového) a dopravní prostředek, který má být využit.

Postup algoritmu

- Prvním krokem algoritmu je nalézt nejbližší body v grafu (odpovídající dopravnímu prostředku) k počátečním a koncovým souřadnicím (Počítá se, že k těmto bodům se člověk dokáže dostat od zadaného místa).
- Dále je na řadě algoritmus A*, který nalezne nejkratší cestu v grafu z počátečního uzlu do koncového. Nejprve kód na obrázku 7.2 prohledá uzly z počátečního uzlu až do koncového.

  Následuje opačný proces, při kterém jsou procházeny uzly od koncového do počátečního podle uložených `perrentId`. Výsledkem je seznam uzlů, kterými vede trasa. A nakonec jsou k uzlům nalezeny i příslušné cesty:
- V případě, že je vybrán jako dopravní prostředek sdílené kolo nebo auto je algoritmus rozdělen do více samostatných částí. Program musí najít cestu od zadaného bodu k sdílenému prostředku pěšky a podle typu prostředku do koncového bodu.

7.4 Výsledné API

Následuje přehled endpointu API servru:

- **GET /cyclePath**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací cyklostezky, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

- **GET /Graph**
  parametry: from: [lng, lot], to: [lng, lot], transportType
  Metoda vrací trasu ze souřadnic from do souřadnic to pro dopravní prostředek definovaný v transportType.
  Návratový kód: 200 OK, 404 Not Found

- **GET /Parking**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací parkomaty, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

- **GET /ZTP**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací parkování pro ZTP, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

- **GET /ZakazStani**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací zákazy stání, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

- **GET /placene_parkovani**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací placené parkování, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

- **GET /parkingLot**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací Parkoviště, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

- **GET /streets**
  parametry: coordinates1: lng, coordinates2: lot, radius
  Metoda vrací úseky ulic, které se nachází do vzdálenosti v okruhu daným souřadnicemi a radiusem.
  Návratový kód: 200 OK, 404 Not Found

```
private async findPath(start: any, end: any, transportType: string) {
    // Inicializuji se 2 prázdné listy.
    const openList = [];  // ukladání otevřených uzlů z kterých probíhá vyhledávání
    const closeList = []; // uzavřené uzly kterými již algoritmus prošel

    start.g = 0; // nastavení hodnoty cesty do startovního budu na nulu
    openList.push(start); // do openlistu je vložen počáteční uzel.
    // poběží dokud není nalezen koncový uzel
    while (true) {
        // Nalezení nejlepšího uzlu mezi otevřenými uzly a následné přesunutí mezi uzavřené
        const currentNodeId = await this.findBestNodeId(openList);
        const currentNode = openList[currentNodeId];
        openList.splice(currentNodeId,  deleteCount: 1);
        closeList.push(currentNode);
        // je nalezený uzel koncový?
        if (String(end._id) === String(currentNode._id)) {
            break;  // ukončí vyhledávání
        }
        // nalezne okolní uzly do kterých vede cesta z currentNode
        const adjacantNodes = await this.findNextNodes(currentNode);
        // cyklus skrz všechny nalezený sousedy
        for (const node of adjacantNodes) {
            // pokud je uzel již uzavřený jde se na dalšího
            if (this.includeNode(closeList, node)) {
                continue;
            }
            // výpočet hodnot uzlu
            // g = nejkratší vzdálenost po cestách z počátečního uzlu
            // h = vzdušná vzdálenos uzlu od koncovéhoo uzlu
            node.g = currentNode.g + await this.pathPrice(currentNode, node, transportType);
            node.h = this.calculateDist(end.coords, node.coords);
            node.f = node.g + node.h;
            node.parrentId = currentNode._id;
            // pokud je jiz v uzel v openlist a zaroven do nej vede
            // kratsi cesta nez v tomto pripade, je uzel zahozen.
            // v opacnem pripade je  pridán do openlistu
            if (this.includeNode(openList, node)) {
                // console.log('includde1');
                // is it better than a current node in openList?
                const foundNode = openList.find((n) => String(n._id) === String(node._id));
                if (node.g > foundNode.g) {
                    continue;
                } else {
                    const index = openList.findIndex((n) => String(n._id) === String(node._id));
                    openList.splice(index,  deleteCount: 1);
                    openList.push(node);
                }
            } else {
                openList.push(node);
            }
        }
    }
```

Obrázek 7.3: Prohledávání grafu

```
let current = closeList[closeList.length - 1];
const route = [];
while (String(current._id) !== String(closeList[0]._id)) {
    route.push(current._id);
    const next = closeList.find((n) => String(n._id) === String(current.parrentId));
    current = next;
}
const links = [];
for (let i = 0; i < route.length - 1; i++) {
    // console.log(route[i]);
    const link = await this.getLink(route[i], route[i + 1]);
    link.properties.transportType = transportType;
    links.push(link);
}
return links;
```

Obrázek 7.4: Zpětný průchod bodů a nalezení cesty

Kapitola 8

Frontend

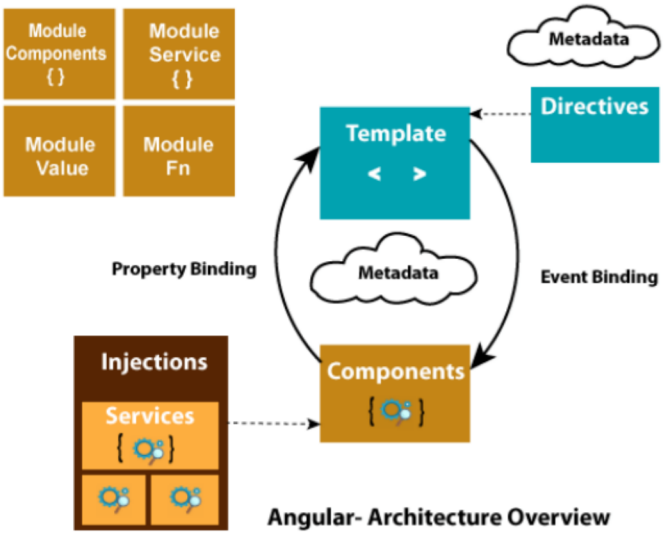
Poslední částí aplikace je klientská část. Jde o webovou aplikaci napsanou za pomocí frameworku Angular, která zajistí komunikaci s uživatelem. Hlavní požadavky na aplikaci jsou:

- Zobrazení různých druhů map.
- Filtrování dopravních prostředků
- Zobrazení dalších prvků souvisejících s dopravou
- Zobrazení vyhledané trasy

8.1 Použité technologie

8.1.1 Angular

Angular je framework pro vývoj single-page webových aplikací. Používá jazyk Typescript, který vychází z JavaScriptu ale rozšiřuje ho o principy OOP. Arcitektura Angularu je založená na komponentách. Každá komponenta je reprezentována třídou obstarávající logiku komponenty a http šablonou a css souborem pro stylování. Vše je propojené pomocí takzvaných direktiv. Webová stránka je složena z těchto komponent, které mezi sebou komunikuji přes služby (service). architektura je znázorněna na obrázku.



Obrázek 8.1: Architektura Angular
(převzato z: https://www.javatpoint.com/angular-7-architecture)

8.1.2 Openlayers

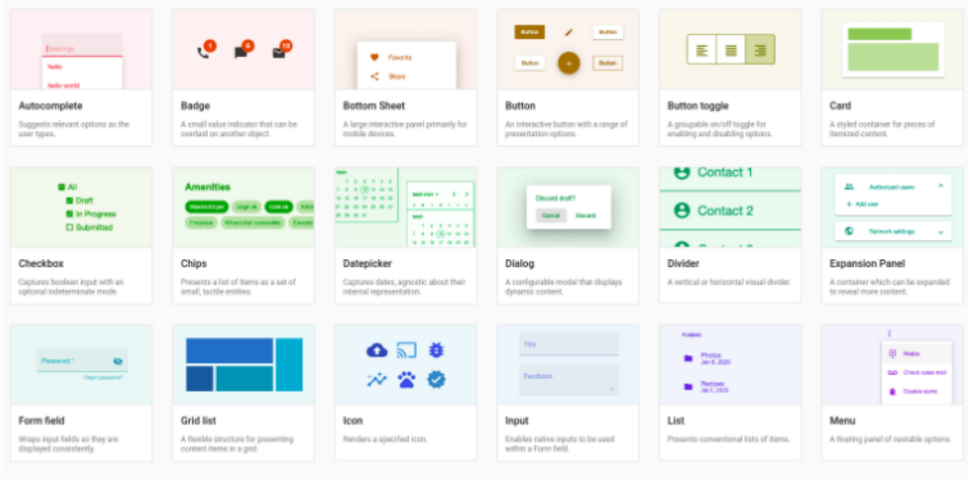
Pro zobrazení mapy a všech ostatních dat je využita knihovna OpenLayers. Knihovnu umožňuje zobrazení mapových podkladů z jakéhokoliv zdroje. Na základní mapu pak mohou být přidávané další vrstvy (layery) s vektrovými daty. Jednotlivé vrstvy mohou být dynamicky přidávány, měněny a nebo odebrány. To umožní v aplikaci měnit základní mapu a filtrovat, které data mají být zobrazena.

Obrázek 8.2: Logo openLayers
(zdroj: https://twitter.com/openlayers)

8.1.3 Angular material

Pro lepší design a jednoduší implementaci je využita knihovna Angular material. Jde o knihovnu napsanou přímo vývojáři Angularu. Knihovna

obsahuje mnoho grafických komponent, které se snadno využiji a tak ulehčí grafický návrh.



Obrázek 8.3: přehled komponent angular material (zdroj: screenshot z https://material.angular.io/components/categories)

8.2 Architektura aplikace

Webová aplikace je složena ze 4 komponent, které mezi sebou komunikují pomocí služeb. Nejdříve jsou popsány služby a jaké data uchovávají. Následně jsou popsány komponenty.

8.3 Služby

Služby (service) jsou využity jako prostředník mezi komponentami pro komunikaci. Každá služba udržuje informace o konkrétních stavech. Další služby budou zmíněny přímo s komponentou, se kterou jsou více spjaty.

8.3.1 BaseMapService

Tato služba obstarává vše okolo základní mapy. Jde o základní vrstvu s mapou. Služba inicializuje TileLayery s různými druhy základních map. Zdrojem dat

jsou OpenStreetMap [ope21]. Služba obsahuje metody na přepínání mezi mapami.

8.3.2 GeoLocationService

Služba na poskytování aktuální polohy uživatele. Při inicializaci vyhledá polohu a následně každých 20s polohu obnovuje. Poloha se ukládá do layeru (vrstvy mapy) tak aby ji bylo možno zobrazit.

8.3.3 GolemioService a RestAPIService

Tyto dvě služby slouží k získání dat z backendu a Golemia API. Obě služby obsahují metodu `getItems`, která vrací typ Observable s požadovanými daty. RestAPIService má navíc metodu, která pošle dotaz na vyhledání trasy mezi zadanými body.

8.3.4 RoutePlanningService

Služba, která zprostředkovává komunikaci s předchozí zmíněnou službou. Kromě toho data vkládá do vrstvy aby se mohla zobrazit na mapě. data jsou složena ze zadaného koncového a počátečního bodu a nalezené cesty.

8.3.5 LayerControllerService

tato služba inicializuje pole vrstev. Každá vrstva obsahuje data o jednom konkretním druhu zobrazovaných prvků (aut, míst na parkovaní atd.). Pro každou vrstvu je vytvořena instance třídy `Layer`, která obstarává vytvoření vrstvy a načítání dat.

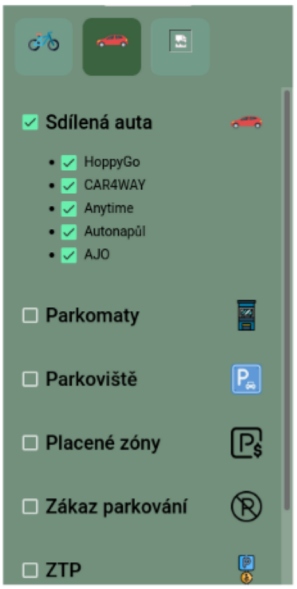
8.4 Komponenty

Komponenty jsou hlavními stavebními prvky aplikace. Každá komponenta se skládá z třídy, která se stará o logiku, html šablony a css pro stylování.

8.4.1 sideMenu komponenta

Komponenta, která zobrazuje jaké prvky jsou možné zobrazit na mapě. Obsahuje 3 módy: kolo auto a hromadná doprava. Mezi těmito módy je možné přepínat pomocí horní lišty s tlačítky zobrazující druhy dopravy. Každý druh má prvky, které jsou možné v mapě zobrazit. Pomocí Checkboxu je možné tyto věci přidávat a odebírat.

změny provedené v komponentě jsou propagované do služby sideMenuService. Tato služba má proměnné filterState a mapMode typu Subject. Při každé změně parametrů v komponentě je v této službě volána metoda next() na subjectu, který o změně informuje všechny odběratele (Subscribers) v jiných komponentách a službách.



Obrázek 8.4: Komponenta sideMenu

8.4.2 Search komponenta

Komponenta přes, kterou se zadávají počáteční a koncový bod pro vyhledání trasy. Tlačítko vyhledat zavolá metodu `findRoute` ve službě `routePlanningService`

Obrázek 8.5: Komponenta search

8.4.3 MapType komponenta

Komponenta se stará o přepínání základní mapy. Po kliknutí na ikonu mapy se zobrazí druhy základních map. Při výběru druhu mapy se nastaví nový druh mapy ve službě baseMapService a stejným mechanizmem jako u předchozích komponent se distribuuje nová hodnota mezi všechny odběratele.

Obrázek 8.6: Ikona pro zobrazení druhů základních map

Obrázek 8.7: Ikony pro výběr základní map

8.4.4 ol-map

Základní komponenta pro zobrazení mapy. V komponentě je vytvořena instance mapy, která bude udržovat všechny data zobrazená na mapě a základní mapový podklad. Data jsou do vkládána ve vrstvách. Vrstvy pro zobrazení prvků jsou uloženy ve službě `LayerControllerService`. Vrstva pro zobrazení trasy je v `RoutePlanningService` a pro uživatelovi pozice v `GeoLocationService`.

8.4.5 overLayBox

Tato komponenta schovaná uvnitř mapy a obstarává vyskakovací dialogy. Po kliknutí na ikonu nebo útvar, který reprezentuje prvek v mapě zobrazí tato komponenta detail o prvku

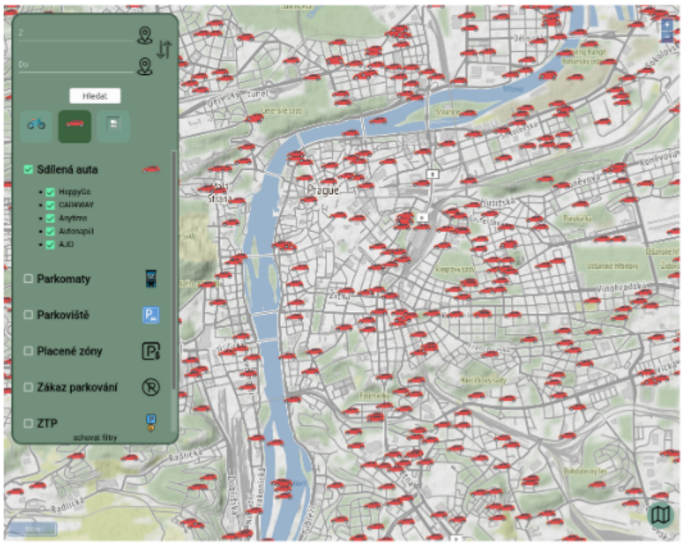
Obrázek 8.8: Zobrazení detailu u sdíleného auta

8.5 grafický návrh aplikace

Aplikace je navržena tak aby jí bylo možno lehce používat jak na desktopu tak i na mobilním telefonu. zobrazení jednotlivých komponent se tedy liší při různých velikostech displeje. zobrazení budou 2 a to jedno pro obrazovku širší než vyšší (desktopové verze) a druhé naopak (mobilní verze).

Základním prvkem je mapa, která vždy pokrývá celou obrazovku. Ostatní prvky se následně zafixují na danou vzdálenost od okrajů obrazovky. Menu s vyhledáváním a filtrem dopravních prostředků je umístěno v případě deskto-

pové verze do levého horního rohu a u mobilní verze zakrývá celý horní okraj. Komponenta `mapType` je umístěna vždy a levém dolním rohu.



Obrázek 8.9: screenshot celé aplikace při zobrazení sdílených aut

8.5.1 Ovládání

Aplikace je udělána tak aby byla co nejjednodušeji ovladatelná. Mapu může uživatel přibližovat, oddalovat a posouvat. Boční menu slouží k filtraci dopravních prostředků zobrazených v mapě. Při kliknutí na mapu se zobrazí možnost z nebo do daného místa naplánovat cestu. Po vybrání obou bodů je možné kliknout na tlačítko vyhledat.

Část III

Nasazení a Testování aplikace

Kapitola 9

Nasazení aplikace na server

Ve chvíli kdy je kód připraven, může být aplikace nasazena na server. Ve druhé kapitole již bylo popsáno jak spustit databázi v cloudu a tak je to v této části vynecháno. Pro Angular aplikaci a node server bude potřeba zprovoznit server, na kterým mohou běžet. Jako server byla vybrána nejnovější verze distribuce Ubuntu Server 21.04 [CL21].

9.1 Instalace Ubuntu Serveru

Ubuntu server je volně dostupný na stránkách Ubuntu. Při výběru "manual server instalation"je stažen iso obraz systému. Iso soubor se dále nahraje na server a při startu se do něj nabootuje. Během instalace serveru dochází k základnímu nastavení serveru. Nastaven je jazyk a uživatelský profil, síť atd.

Po dokončení instalace serveru je nutné zabezpečit připojení. Na to se dá využít program *ufw* [Št10], přes který se dá nastavit firewall. K serveru bude možno přistoupit přes http. Této službě odpovídá standardně port 80. Port se povolí pomocí příkazu.

sudo ufw allow http

Pro aplikaci pravidel pak slouží příkaz

sudo ufw enable

9.2 Nasazení Frontendu

Server je v provozu, a je tedy možné na něj nasadit Angular aplikaci. Pro běh webové aplikace je zapotřebí http server. Pro tento účel je využit NGINX [htt21b]. Server se nainstaluje příkazem

```
sudo apt-get install nginx
```

dále se povolí jeho provoz v firewallu příkazem

```
sudo ufw allow 'Nginx HTTP'
```

server se spustí příkazem

```
sudo systemctl start nginx
```

Další krok je build aplikace. To se provede přes příkaz níže spuštěným v kořenovým adresáři kódu aplikace.

```
ng build -prod
```

Příkaz provede build aplikace a výsledek nahraje do složky *dist*. Složku následně překopírujeme do adresáře
var
www
html. Nyní když se v prohlížeči otevře stránka s IP adresou serveru bude tam běžet tato aplikace.

9.3 Nasazení backendu

NodeJs aplikace bude nasazena na stejný server jako klientská aplikace. O její běh se bude starat PM2 . Jde o správce procesů určený pro JavaScript runtime v nodeJs . Nejprve je proveden build aplikace, při kterém jsou soubory napsané v Typescriptu kompilované do JavaScriptu. Aplikace se spustí příkazem `pm2 start app.js` kde `app.js` zkompilovaný soubor `app.ts` z původního zdrojového kódu.

Kapitola 10

Uživatelské testování

Nedílnou součástí vývoje softwaru je testování. Testování odhalí chyby, které mohli být skryté při vývoji. Tyto chyby jsou následně opraveny a aplikace se stává spolehlivější. Při uživatelském testování mohou být odhaleny chyby, které by vývojáři odhaleny nebyly kvůli rozdílnému pohledu na věc. Uživatelé testují software na základě předpřipravovaných scénářů. Uživatel se snaží aplikací projít podle návodů a všímá si chyb či nesrozumitelných věcí v aplikaci. Připomínky následně sdělí vývojářům, kteří danou chybu nebo nesrozumitelné chování aplikace mohou opravit.

10.1 Průběh testování

Testování probíhá za pomocí mnoha dobrovolníků, kteří jsou potencionální uživatele aplikace. Uživatele prochází jednotlivé scénáře, přičemž si volí libovolně místa pro testování. Tím dojde k testování více variant stejného scénáře. Uživatele si jednotlivé problémy a chyby v aplikaci zaznamenali. Zaznamenali taktéž postřehy kdy aplikace nebyla příliš srozumitelná nebo se jim nelíbil design. Uživatelé také mohli poznamenat jakoukoliv jinou věc, které sice přímo nesouvisí s daným scénářem ale nefungovala korektně.

10.1.1 Scénáře

Prvním krokem k otestování aplikace jsou tedy testovací scénáře. Tyto scénáře by měli vycházet z reálných situací jak je aplikace zamýšlená používat. Scénáře byly navrženy celkem 4:

- **Scénář č. 1.** Uživatel hledá ve svém okolí sdílený dopravní prostředek. Na typu prostředku nezáleží. Aplikaci chce vyžít aby mu ukázala jeho současnou polohu a ukázala nejbližší dopravní prostředky. Následně si jeden vybere a aplikace mu ukáže cestu k němu.
- **Scénář č. 2.** Uživatel hledá cestu ze své současné pozice na místo, které vybere na mapě. Účelem je srovnat různé cesty (Za využití různých dopravních prostředků).
- **Scénář č. 3.** Uživatel hledá na dané pozici na mapě možnosti parkování. Chce zobrazit jak místa na parkování pro auto tak pro kola.
- **Scénář č. 4.** Uživatel plánuje cestu z místa A do místa B na kole, chce aby mu aplikace ukázala jak cestu na vlastním kole ze současné pozice tak i cestu za použití sdíleného kola v okolí aktuální polohy.

10.1.2 Potencionální Uživatelé

Pro testování je nutné vybrat takové lidi, pro které je daná aplikace určena. Tím se zajistí že je testování relevantní.

Pro tuto aplikaci jsou bráni jako potenciální uživatelé mladí lidé kteří jsou více otevření používání sdílených dopravních prostředků.

10.1.3 Výsledky testování

Aplikace byla otestována celkem 6 uživateli. Uživatelé byly vybráni ze skupiny potenciálních uživatelů. Zároveň byl kladen důraz aby nebyly ze stejné skupiny.

Při testování Našli tyto problémy, které jsou zařazeny do následujících skupin.

- **Design** Objevily se chyby v designu. Konkrétně šlo o špatné načítání některých ikon v bočním menu. V bočním menu se taktéž zobrazovalo špatně možnost schovat filtry a této možnosti si uživatelé ani nevšimli. Posledním designovým problémem, na který uživatele upozornili byla velmi malá ikona pro změnu základní mapy, které si většina ani nevšimla.
  Řešení chybějící ikony byly doplněny a špatně viditelné prvky zvýrazněny.

- **Ovládání aplikace** Ovládání hodnotili uživatele jako jednoduché, ale ze začátku zmatené. Nebylo nikde řečeno jak aplikaci ovládat.
  Řešení Tento problém se neřešil jelikož po chvilce používání uživatel pochopí jak aplikace funguje.

- **Hledání trasy 1** Při hledání trasy docházelo k problému příliš dlouhého vyhledávání, které zároveň nebylo nikde znázorněno, že probíhá. Uživatel tak nevěděl jestli se trasa skutečně vyhledává.
  Řešení Chování bylo upraveno tak, že po stisknutí tlačítka hledat tlačítko zmizí a objeví se nápis načítání. Tlačítko se znova objeví až když se cesta najde.

- **Hledání trasy 2** Dalším problémem byl v samotné trase, která někdy vedla v protisměru.
  Řešení Úkázalo se, že při implementaci algoritmu byl opomenut tento faktor a algoritmus byl tedy doplněn o tuto funkcionalitu

- **Hledání trasy 3** Trasa byla někdy vedena objektivně delší trasou než je možné
  Řešení Problém se nacházel ve špatném přepočtu vzdáleností hran v algoritmu

- **Nezobrazení sdílených kol** Po zaškrtnutí checkboxu na zobrazení sdílených kol se nezobrazilo žádné kolo
  Řešení Důvodem této chyby bylo, že v portálu Golemio byly vymazána data o sdílených kolech. Volba pro zobrazení kol však zůstala pro případ, že se do portálu data opět vloží.

Přestože aplikace prošla tímto testování je velmi pravděpodobné, že některé chyby zůstaly schované a projeví se až při využití aplikace více uživateli po delší dobu.

Kapitola 11

Limity aplikace

Druhým typem testování bude testování zatížení aplikace. V uživatelském testování se odhalilo spoustu chyb v aplikaci. Testování však neřešilo zatížení aplikace a jak bude fungovat v případě, že se připojí více uživatelů. Úkolem následujícího testu bude odpovědět na otázky kolik uživatelů je aplikace schopna obsloužit tak aby nepoznali nic na jejím fungování. Díky tomu bude možné do budoucna odhadnout jaký je zapotřebí hardware aby byla aplikace provozuschopná za určitých podmínek.

Testovat aplikaci jakou zátěž zvládne je možné více způsoby. Nejjednodušším způsobem je přímo manuální testování. To má však problém v tom, že se nedá přesně opakovat. Druhou možností je využít nástroje, které jsou na toto testování dělané.

Testování ukázalo, že při dotazech na data z databáze není server téměř vytížený. Mnohonásobně náročnější je však dotaz na vyhledání trasy. Vyhledávání je náročné především na paměť. Algoritmus si udržuje v paměti procházené body, kterých exponenciálně přibývá s délkou trasy. Při testech docházelo k zabrání až 100MB operační paměti. To dává přibližný odhad, kolik dotazů na vyhledání trasy je server s určitým přiděleným množstvím operační paměti schopen obsloužit.

Kapitola 12

Závěr

Lidé jsou zvyklí používat jeden konkrétní dopravní prostředek na pohyb po městě a obvykle je ani nenapadne využít jiný i když je rychlejší a ekonomičtější. Vytvořená aplikace se toto snaží řešit tím, že nabídne srovnání různých typů dopravy. Kromě plánování cest zobrazí i další informace dopravního charakteru, jako jsou například informace o možnostech parkování. Celá aplikaci vychází z dat, které jsou volně k dispozici na serverech Prahy. Tyto data Praha zveřejnila právě proto, aby byla dále využita a pomáhala zkvalitnit služby v Praze. Aplikace je důkazem, že tyto data mají ekonomický potenciál a Praha tak neplýtvala veřejnými financemi na jejich zveřejnění, ba naopak podpořila tím vývoj nových služeb.

Aplikace sice dokáže naplánovat cestu různými dopravními prostředky, ale není schopna je kombinovat. To představuje možné budoucí rozšíření aplikace. Služby, které by dokázaly člověku vyhledat trasu kombinovanou z několika různých dopravních prostředků zatím fungují pouze omezeně, a tak jsou vidět v dalším rozvoji těchto služeb ještě rezervy. Rezervy jsou i ve využívání sdílených dopravních prostředků, jejichž provozování má dobrý efekt na obsazenost parkovacích míst ve městech.

Přílohy

Příloha A

Literatura

[AWS21] https://aws.amazon.com/ Amazon Web Services, Inc. or its affiliates, *Amazon web services*, 2021.

[cit] *World geodetic system (zkratka wgs84), dostupné z: https://training.gismentors.eu/open-source-gis/soursystemy/wgs84.html.*

[cit99] *Zákon č. 106/1999 sb., o svobodném přístupu k informacím*, 1999.

[CL21] https://ubuntu.com/download/server Canonical Ltd, *Ubuntu. enterprise open source and linux*, 2021.

[dvpp] Otevřená data v ČR: Portál pro poskytovatele, *Co jsou otevřená data, https://opendata.gov.cz/informace:start.*

[GIS14] GISMentors, *Mercatorovo zobrazení — Školení Úvod do (open source) gis*, 2014.

[Gro21] The World Bank Group, *Starting an open data initiative, http://opendatatoolkit.worldbank.org/en/starting.html*, 2021.

[htta] Microsoft https://azure.microsoft.com, *microsoft azure.*

[httb] Google Cloud. https://cloud.google.com, *Cloud computing services.*

[htt21a] https://www.geoportalpraha.cz, *Hledání dat, služeb a map*, 2021.

[htt21b] https://www.nginx.com/, *High performance load balancer, web server, reverse proxy. nginx*, 2021.

[IparhmP] https://www.iprpraha.cz/clanek/1313/otevrena-data-open-data Institut plánování a rozvoje hl. m. Prahy, *Otevřená data / open data.*

[Mon] Inc. https://www.mongodb.com/ MongoDB, *The most popular database for modern apps.*

[Mon21] Inc. MongoDB, *Mongodb atlas, https://www.mongodb.com/cloud/atlas*, 2021.

[OI18] a. s. Operátor ICT, *Golemio - katalog datové platformy*, 2018.

[ope21] openstreetmap, *https://www.openstreetmap.org*, 2021.

[Roy19] Baijayanta Roy, *A-star (a*) search algorithm*, 2019.

[Soec21] IBM StrongLoop and other expressjs.com contributors, *Express - node.js web application framework.*, 2021.

[TA] Christopher Williams Thaddeus Abiy, Hannah Pang, *Dijkstra's shortest path algorithm.*

[VK] M. Konečný Z. Stachoň-K. Tajovská : V. Kaplan, K. Keprtová, *Multimediální učebnice kartografie a geoinformatiky.*

[Št10] Adam Štrauch, *Ufw: firewall jednoduše a rychle, https://www.root.cz/clanky/ufw-firewall-jednoduse-a-rychle*, 2010.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kostohryz** Jméno: **Martin** Osobní číslo: **457183**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Plánování trasy s pomocí otevřených dat

Název diplomové práce anglicky:

Route planning using open data

Pokyny pro vypracování:

Seznamte se s otevřenými datovými rozhraními, poskytovanými v rámci platformy 'Opendata Praha'. Soustřeďte se zejména na kartografická data (geoportal) a informace o sdílených dopravních prostředcích (golemio).
Navrhněte architekturu webové aplikace, která uživateli dovolí plánovat trasu s ohledem na využití sdílených dopravních prostředků, tj. dovede cestujícího ze zadaného bodu k takovému sdílenému prostředku, který si vybere. Popište míru závislosti na službách třetích stran a zmíněnou aplikaci naimplementujte.
Klíčové požadované funkce aplikace:
- plánování trasy (představte dostupná řešení poskytovaná formou JS API)
- filtrace druhů sdílených dopravních prostředků
- přepínání mapových podkladů v kontextu vybraného prostředku
- využití služby geolokace pro výběr zdrojových souřadnic
Aplikaci zveřejněte a podrobte kvalitativnímu uživatelskému testování. Popište limity (datové, frekvenční) využívaných služeb a odhadněte míru zátěže, kterou je aplikace schopná obsloužit.

Seznam doporučené literatury:

Michael Dorman, Introduction to Web Mapping; 2020 Chapman and Hall / CRC Press; ISBN 9780367861186
https://developer.mozilla.org/en-US/docs/Web
https://opendata.praha.eu/
https://golemioapi.docs.apiary.io/#reference/traffic
https://www.geoportalpraha.cz/cs/data/otevrena-data/seznam

Jméno a pracoviště vedoucí(ho) diplomové práce:

RNDr. Ondřej Žára, Katedra počítačové grafiky a interakce

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

Datum zadání diplomové práce: **28.01.2021** Termín odevzdání diplomové práce: **21.05.2021**

Platnost zadání diplomové práce: **30.09.2022**

RNDr. Ondřej Žára	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. _____
Datum převzetí zadání

_____
Podpis studenta