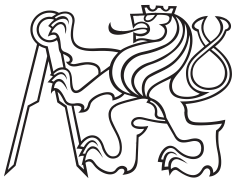
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Odpadní kontejnery

Jakub Zíka

Vedoucí: RNDr. Ondřej Žára
Obor: Otevřená informatika
Studijní program: Softwarové inženýrství
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zíka** Jméno: **Jakub** Osobní číslo: **483798**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Odpadní kontejnery

Název bakalářské práce anglicky:

Waste containers

Pokyny pro vypracování:

Prozkoumejte dostupná data o odpadních kontejnerech z veřejného datasetu 'opendata.praha.eu'. Na základě těchto dat navrhněte a naimplementujte webovou aplikaci, která bude poskytovat dvě hlavní funkce:
1) geo-lokalizované vyhledávání kontejnerů
2) odhad zaplněnosti konkrétního kontejneru
Aplikace bude primárně zaměřena na mobilní uživatele (tj. bude mít rysy PWA), ale fungovat bude i na desktopových počítačích a prohlížečích. Diskutujte různé varianty odhadu zaplněnosti konkrétního kontejneru v konkrétní čas na základě technologických možností (senzory zaplnění, crowdsourcovaná data, statistické odhady).
Vzniklou aplikaci podrobte kvalitativnímu uživatelskému testování a vystavte ji na veřejně dostupné webové adrese. Zhodnoťte použité API třetí strany a navrhněte jeho případné vylepšení, s ohledem na neúplná či problematicky získatelná data. Nezapomeňte zdokumentovat použitá klientská API a knihovny nutné pro realizaci aplikace.

Seznam doporučené literatury:

tps://opendata.praha.eu/
https://web.dev/pwa-checklist/
https://www.geoportalpraha.cz/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ondřej Žára Katedra počítačové grafiky a interakce

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

Datum zadání bakalářské práce: **15.02.2022** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **19.02.2024**

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

III. PŘEVZETÍ ZADÁNÍ

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

. _____ Podpis studenta

Datum převzetí zadání

Poděkování

Děkuji svému vedoucímu tohoto projektu RNDr. Ondřeji Žárovi za podporu při tvorbě této Bakalářské práce

Prohlášení

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

V Praze, 20. května 2022

Abstrakt

Předmětem této práce je predikce plnosti kontejnerů na tříděný odpad v Praze. A webová aplikace, která umožní uživateli snadné vyhledání kontejneru na tříděný odpad v jeho okolí.

Klíčová slova: kontejnery na tříděný odpad, odhad plnosti, webová aplikace, otevřená data, geospaciální datová analýza

Vedoucí: RNDr. Ondřej Žára

Abstract

This bachelor thesis focuses on predicting fullness of sorted waste containers. And web aplication, that allows users to easily search for sorted waste containers in their area.

Keywords: sorted waste containers, fullness prediction, web application, open data, geospatial data analysis

Title translation: Waste containers

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

V Praze se nachází 3,400 veřejnosti přístupných stanovišť s kontejnery na tříděný odpad. Město čelí težkému problému rozhodování, kde mají být stanoviště umístěna, jaké druhy odpadu na nich lze vyhodit a jak správně nastavit svozovou frekvenci. Ne vždy se tento úkol podaří vyřešit, a nastává situace, kdy kontejnery bývají přeplněné. Lidé tak buď přinesený odpad nechají vedle popelnice, čímž přidělávájí práci popelářům, hledají jiný, snad prázdný, kontejner, a nebo se s odpadem vrací domů a doufají že za pár dní se jim odpad vynést povede. Dalším, možná i vážnějším, problémem je neinformovanost občanů, kde se nachází jaké kontejnery na tříděný odpad.

Kapitola 2

Otevřená data a Pražské iniciativy

Digitalizace přichází do všech částí našeho života. Nejen do života osobního, ale stejně tak si nachází cestu i do provozu státních institucí. Problém s přehnaným množstvím papírů nahrazuje problém s velkým objemem dat. Instituce s nimi musí dokázat správně nakládat a neví, s jakými daty nakládají ostatní. A když už data dostanou, mohou být v uplně jiném formátu, a tedy musí vynaložit větší úsilí, aby s nimi dokázali efektivně pracovat.

Na tento problém se zaměřuje projekt Evropské unie *OPZ č. CZ.03.4.74/0.0/0.0/15_025/0004172*[19], jehož předmětem je rozvíjení oblasti otevřených dat v prostředí veřejné správy a implementace mezinárodních standardů. Díky němu vznikl **Národní katalog otevřených dat** [18], který agreguje data ze státních institucí. Důsledkem toho je posíleno budování e-govermentu a je umožněna snažší mezinárodní spolupráce. V Národním katalogu se momentálně (květěn 2022) nachází 141,184 volně přístupných datových sad od 246 poskytovatelů.

Specificky pro Prahu existuje podobný katalog **Opendata Praha** [17], který je zřízen magistrátem Hl.m. Praha.

Dalším, více specifickým, pražským katalogem je **GeoPortál Praha** , který se zaměřuje na agregaci geografických data a jeho zřizovatelem je *Insitut plánování a rozvoje hlavního města Prahy* (IPR Praha). V portálu se nachází jak data ostatních pražských institucí, tak data, která vytváří samotný IPR Praha.

Dále existuje datová platforma Golemio[3] vytvořena společností Operátor ICT a.s.[5]. Její funkcionalita je popsána následovně: *"Datová platforma je aplikační set pro poskytování sady služeb nad městskými daty - shromáždění dat z řady zdrojů a poskytovatelů (městské společnosti, externí systémy, OpenData, statická data), zpracování, transformace a konsolidace dat, datová analýza a následné poskytnutí výstupů a prezentace dat svým klientům, jak ve formě datových analýz, vizualizací, publikování OpenDat, nebo poskytování otevřeného realtime API."* [20, převzato ze stránkek s dokumentací platformy]

(a) : Podzemní kontejnery

(b) : Kontejner se spodním výsypem.

Obrázek 2.1: Ukázka kontejnerů

2.1 Smart City Prague

Koncepce Smart Prague[4], kterou vytvořili Magistrát hl.m. Praha a Operátor ICT[5], si dává za cíl zlepšení fungování města za pomoci digitalizace. Stanovuje konkrétními cíle, kterými se zaobírá a navrhuje projekty vybraným institucím. Cíle jsou následující: zlepšení mobility občanů, chytré budovy a energie, **bezodpadové město**, atraktivní turistika, lidé a městské prostředí a **datová oblast**.

Díky této koncepci vznikly pro tuto bakalářskou práci dva důležité projekty:

2.1.1 Chytrý svoz odpadu

Prvním z nich je projekt "Chytrý svoz odpadu"[21]. Jeho cílem je instalace senzorů do kontejnerů zapuštěných do země a těch se spodním výsypem. Svoz těchto kontejnerů je finančně a časově náročnější kvůli potřebě specifického popelářského auta. Data o plnosti kontejnerů pomáhají městu a svozovým společnostem optimalizovat svozovou frekvenci, aby se předešlo zbytečně častým svozům, a naopak mohly zvýšit frekvenci u stanovišť, která jsou často přeplněná. Pilotní fáze projektu proběhla a byla vyhodnocena jako úspěšná a již přechází do plné implementace. Výsledkem pilotní části je 460 kontejnerů obohacených o senzor na měření plnosti. Získaná data jsou přístupná na platformě Golemio (včetně historických dat o plnosti). Po realizaci projektu budou tyto senzory nainstalovány do 6000 kontejnerů.

2.1.2 RFID odpadové nádoby

Druhým projektem Smart Prague je "RFID odpadové nádoby"[22]. Město nemá přehled o tom, kdy jsou kontejnery vyváženy, frekvence svozu je uváděna jen orientačně, a navíc se může měnit. Cílem pilotní fáze tohoto projektu

je osázet 1000 kontejnerů na tříděný odpad s horním výsypem RFID čipy a přidat RFID čtečky na popelářská auta. Při vyprázdnění popelářské auto přečte kód kontejneru a automaticky odešle data o provedeném svozu. Tím bude moct město kontrolovat, zda ke svozům skutečně dochází. Projekt se nachází ve fázi příprav, a tudíž nemá dostupné žádné výsledky.

Kapitola 3

Datová analýza

V této kapitole vysvětluji co jsou geospaciální data a jak se s nim pracuje. Dále zminuji jak a v jakém prostředí jsem prováděl datovou analýzu a jak jsem se snažil přiblížit možnosti predikce plnosti kontejnerů na tříděný odpad.

Geospaciální data jsou taková data, která mají k sobě asociovanou geografickou složku. Tato data tedy mohou popisovat vlasnost objektů na zemi (lokace, oblasti) a dají se zjištovat různé prostorové vazby mezi nimi.

Datová analýza je proces získávání, čištění, transformování a modelování dat za účelem získávání nových přínosných poznatků. Geospaciální datová analýza tento proces rozšiřuje o práci s geospaciálními daty.

Obrázek 3.1: Ukázka geospaciálních dat(poloha vstupu do metra[23])

3.1 Používané souřadnicové systémy

Aby bylo možné pracovat s daty, které popisují reálné objekty, je potřeba systému k určování jejich polohy na zemi. Pro tento účel byly vytvořeny souřadnicové systémy (coordinate reference system).

Zeměpisné souřadnicové systémy (GCS) používají zeměpisnou výšku, délku a nadmořskou výšku ve vztahu k referenčnímu tělesu. Systém je určen tímto referenčním tělesem (koule nebo sféroid) a sití přesných bodů na zemi, které "ukotvují" těleso na zeměkouli.

Použitím vybraného kartografického zobrazení na zěmepisný souřadnicový systém se dostaneme do **souřadnicového systému rovinných souřadnic (PCS)**. Tímto docílíme zobrazení tří dimenzionálního objektu na dvourozměrnou plochu. Toto zobrazení je sice zatíženo zkreslením, ale za to se s daty v tomto formátu snáze pracuje a umožnuje data vizualizovat na ploše (tištěné mapy) [50].

V České republice jsou data dostupná nejčastěji ve dvou formátech. Prvním z nich je**WGS84 (World geodetic system 1984)**[48], který patří mezi zeměpisné souřadnicové systémy a je nejčastěji používáným formátem na světě. Druhým je **S-JSTK (Systém jednotné trigonometrické sítě katastrální)**[49], který patří mezi souřadnicové systémy rovinných souřadnic. Byl vytvořen pro zeměměřičské účely České republiky a Slovenska.

3.2 Reprezentace geospaciálních dat

Reprezentace dat dělíme podle druhu informace, kterou zachycují. Nejčastěji se používá reprezentace vektorová či rastrová.

3.2.1 Vektorová

V případě vektorové reprezentace jsou geometrické objekty tvořené body v daném souřadnicovém systému. Základním stavebním kamenem těchto objektů jsou vektory.

Nejčastějšími objekty jsou:

- Body (př. poloha dopravního značení)
- Úsečky (př. tok řeky)
- Polygony (př. tvar městské části)

Díky jejich matematické definici se dají snadno zjišťovat vlastnosti jednoho objektu, ale i vztah více objektů.

Data jsou nejčastěji ukládána ve formátu Shapefile[42] a GeoJSON[43].

3.2.2 Rastrová

Rastrová reprezentace pracuje s pixelovanými daty, kde každý pixel reprezentuje specifickou lokaci na mapě. Lze je chápat jako obrázek se spaciální vazbou, která jej asociuje s danou lokací. Příkladem jsou satelitní snímky země a výšková mapa[24].

3.3 Programovací jazyk Julia

Pro datovou analýzu jsem se rozhodl použít jazyk Julia. Julia je dynamicky typovaná, využívá JIT kompilace (Just in time compilation) a slouží pro vědecké účely. Její první verze byla vydaná v roce 2012[30]. Kvůli relativně krátké době její existence nemá tak velký ekosystém jako ostatní jazyky (Python, C, R). Tento problém řeší možností volat funkce z cizích jazyků.

Část analýzy jsem prováděl v notebooku Pluto[**?**]. Jedná se o prostředí, které vychází z literárního programování od Donalda E. Knutha[44]. Skládá se z bloků, kde každý blok buď může obsahovat spustitelný kód, a nebo text, který slouží pro čtenáře a vysvětluje, co se v jakých blocích děje. Umožňuje interaktivní vývoj, který je vhodný pro datové analýzy.

3.4 Operace s vektorovými daty

Zde zmíním operace, které lze dělat s vektorovými daty v knihovně (ArchGDAL.jl[31]), kterou jsem pro práci s těmito daty použil. Tato reprezentace slouží pro zjednodušení popsaných operací, které jsem prováděl s geospaciálními vektorovmi daty a nemusí být matematicky korektní.

- $Body$ - množina všech možných bodů
- $Polygony$ - množina všech možných polygonů
  kde body jsou explicitně definovány v datové sadě a slouží pro jednodušší reprezentaci dat kdy není třeba pracovat s jejich Polygonální reprezentací.

A následovné operace s těmito daty

- $\in: Body \times Polygony \Longrightarrow \{true, false\}$
  predikát, který zjišťuje jestli bod leží uvnitř polygonu
- $centroid : Polygony \Longrightarrow Body$
  získá těžiště Polygonu
  neplatí $\forall p \in Polygony, centroid(p) \in p$
- $pointOnSurface : Polygony \Longrightarrow Body$
  vrátí bod ležící uvnitř polygonu
  platí $\forall p \in Polygony, pointOnSurface(p) \in p$
- $area : Polygony \Longrightarrow \mathbb{R}^+$
  vrátí plochu polygonu
- $distance : Body \times Body \Longrightarrow \mathbb{R}^+$
  vrátí vzdálenost dvou bodů, za použití Haversínova vzorce
- $\cap : Polygony \times Polygony \Longrightarrow Polygony$
  vrtátí průnik dvou polygonů
- $\cup : Polygony \times Polygony \Longrightarrow Polygony$
  vrtátí sjednocení dvou polygonů

3.5 Vizualizace dat

Aby bylo možné s geografickými daty pracovat je velmi přínosné je zobrazit na mapě. To jde v Julii přímo v notebooku, a nebo přes externí nástroje. Avšak Julia nenabízí knihovny, s kterými by šlo snadno pracovat a které by zajistily snadné vykreslení vlastních dat na mapách a zároveň zajistily interaktivitu. Je sice možnost volat Pythonovský kód přímo z Julie a tím

pádem využít ekosystému knihoven Pythonu, kde již takové knihovny existují, ale data je potřeba transformovat do formátu s kterým dané knihovny dokáží pracovat. A zároveň je problém, když je potřeba vizualizovat větší objem dat, která jsou náročná na vizualizaci. V tomto případě je znatelné snižován výkon počítače. Co je snadnější, je vizualice GeoJSON souborů v externích programech. Existuje webová aplikace a rozšíření do VSCode Kepler.GL[46] , která umožňuje vizualizovat GeoJSON soubory a obsahuje jednoduché rozhraní, jak kombinovat více datových sad, ovlivnit vizualizovanou vlastnost dat v závislosti na vybraném parametru.

3.6 Data s lokacemi kontejnerů a jejich plností

Nejdůležitějším použitým datasetem jsou lokace kontejnerů na tříděný odpad.

Pro snadný přístup k datům z Golemio platformy jsem si vytvořil v Julii knihovnu, která poskytuje funkční rozhraní pro vytváření požadavků na jejich REST API[1]. A spojí data dohromady, v případě že se nevejdou do jednoho požadavku. Využil jsem následující endpointy, pro které jsem i vytvořil příslušné funkce v knihovně.

- Získání všech stanovišt s tříděným odpadem
  a s kontejnery na stanovišti

  https://api.golemio.cz/v2/sortedwastestations/

- Získání všech měření pro kontejner
  se senzorem s daným identifikátorem

  https://api.golemio.cz/v2/sortedwastestations/measurements/

- Získání všech svozů odpadu pro kontejner
  se senzorem daným identifikátorem

  https://api.golemio.cz/v2/sortedwastestations/picks/

Ze všech zmíněných endpointů jsem stáhl všechna data, co jsou k dispozici, pro usnadnění budoucí práce a abych předešel zbytečně častým požadavkům na Golemio API.

Při získávání stanovišť s kontejnery jsem narazil na chybu platformy Goleimo. Kde všechna stažená stanoviště měla parametr, že jsou volně přístupná. Myslel jsem si, že pro získání stanovišť, která jsou přístupná pouze obyvatelům musím specifikovat filtrační parametr při dotazování na daný endpoint. Avšak při jeho nastavení se mi stále vrátila stanoviště s volnou přístupností a nikoliv soulkormá stanoviště . Díky tomu že Golemio je open source platforma,

[1] způsob jak lze komunikovat se službou na webu

tak jsem v zdrojovém kódu vyhledal implementaci funkce která obsluhuje daný požadavek. Zde jsem zjistil že hodnota daného filtrační příkazu není použita. To by ale neměl být problém, pouze by se vracela všechna stanoviště nezávisle na přístupnosti. Po dalším pátrání jsem zjistil, že server stáhne data o stanovištích z databáze a následně při převodu do datové struktury sloužící pro klienta přepíše přístupnost u všech stanovišt na "volně dostupné". Vytvořil jsem bug report o chybě na GitLab, kde je uložen kód platformy Golemio. Zde jsem chybu popsal, zadal jak lze chybu reprodukovat a poukázal na příslušné části kódu kde se chyba vyskytuje.

Kód 1 Úsek kodu z Golemio platformy [26]. K přepsání dostupnosti dochází na řádcích **12 - 16**

```
1  for (const station of Object.keys(stationsByCode)) {
2   data.features.push({
3    geometry: {
4     // FIXME code smell
5     // @ts-ignore
6     coordinates: [
7     +stationsByCode[station].longitude || null,
8     +stationsByCode[station].latitude || null
9     ],
10    type: GeoCoordinatesType.Point,
11   },
12   properties: {
13    accessibility: {
14     description: "volne",
15     id: 1,
16    },
17    containers: stationsByCode[station].containers,
18    district: stationsByCode[station].district,
19    id: stationsByCode[station].id,
20    is_monitored: this.hasMonitoredContainer(
21    stationsByCode[station].containers
22    ),
23    name: stationsByCode[station].address || "",
24    station_number: station,
25    updated_at:
26    stationsByCode[station].created_at ||
27    stationsByCode[station].updated_at,
28    knsko_id: stationsByCode[station].knsko_id,
29   },
30   type: "Feature",
31  });
32 }
```

Existuje stejný dataset o stanovištích na tříděný odpad, který je přístupný

na GeoPortálu Praha. Který by jsem mohl použít namísto toho z Golemia. Ale čelil bych zde problému, že data na geoportálu nejsou natolik aktuální a mohl by nastat problém, že reálné stanoviště na tříděný odpad se již změnilo. Rozhodl jsem se tedy tyto dva datasety sloučit za použití identifikátoru názvu stanoviště(adresy). Stanovištím, pro která se nepovedla zjisti přístupnost byla nastavena na neznámou.

3.7 Predikce plnosti

Plán predikce: získat příznaky pro každé stanoviště zjistit korelaci a viabilitu navrhnout regresní funkci s parametry přetvořit příznaky v parametry pro neznámé parametry regresní funkce

3.8 Regresní funkce

"Regrese je modelování funkční závisosti nějaké proměnné na jiné proměnné. Modelujeme závislost proměnné $y \in \mathbb{R}$ na proměnné $x \in X$ (kde X je libovolná množina) regresní funkcí $y = f(x, \theta)$, která je známa až na parametry $\theta \in \mathbb{R}^n$." [1, kapitola 5.1.3].

Hledal jsem funkci, která dokáže popsat plnění kontejneru ve svozovém období. Zvolil jsem regresní funkci

$$f(x, r) = min(1, r \cdot x)$$

Známý parametr x je doba od posledního svozu odpadu, $r = \theta$ je neznámý parametr rychlosti plnění specifický pro každý kontejner. Parametr r lze zjistit pro měřená stanoviště ale u neměřených schází. Zaměřil jsem se na to jestli je možné ho odhadnout z dat Prahy která jsou volně k dispozici.

3.9 Odhad rychlosti plnění

Chtěl jsem otestovat hypotézu, že na čim větší oblast lidí spadá kontejner na daný druh odpadu, tím rychleji se může plnit. Rozhodl jsem vycházet z myšlenky voronoi diagramu a rozšířit ji a přizpůsobit ji pro mé účely. Voronoi diagram je způsob rozdělení plochy pomocí n generujících bodů na n konvexních polygonů. Tak, že každý polygon obsahuje jeden z generujících bodů. A pro každý bod v polygonu platí, že nejbližší generující bod pro něj se nachází ve stejném polygonu[27]. V mém případě jsou generujícími body veřejná stanoviště na veřejný odpad a ostatními body budovy v Praze. A pro měření vzdálenosti slouží délka cesty mezi těmito body ve váženém grafu. Kde grafem je sít pěších tras v Praze a váhami je vzdálenost mezi uzly.

3.9.1 Práce s daty z katastrálního úřadu

Pro informace o všech budovách v Praze jsem použil data z katastrálního úřadu. Ten umožňuje stáhnout data pouze pro dané katastrální území a poskytuje je pouze ve formátu shapefile se souřadnicovým systémem S-JSTK [6]

Ze stránek lze stahovat data pro jednotlivá katastrální území zvlášť. Musel jsem tedy nejdříve zjistit, jaká katastrální území se v Praze nachází a následně pomocí skriptu v Julii stáhl data pro příslušná katastrální území. Data jsem pomocí příkazu **ogr2ogr** převedl ze souřadnicového systému S-JSTK do systému WGS84 a z formátu Shapefile do GeoJSONu. Data pro každé katastrální území obsahovaly: budovy(body a polygony),parcely(body a polygony),tvar katastrlního území(polygon)

Kód 2 Příkaz pro převody formátu ESRI shapefile do GeoJSON, kód pro převod souřadnicových systému je ve formátu pro knihovnu Proj4[**?**]

```
1  ogr2ogr -s_srs "+proj=krovak +lat_0=49.5 +lon_0
    =24.83333333333333 +alpha=30.28813972222222 +k=0.9999+x_0
    =0 +y_0=0 +ellps=bessel +pm=greenwich +units=m +no_defs +
    towgs84=570.8,85.7,462.8,4.998,1.587,5.261,3.56" -t_srs "
    epsg:4326" -f GeoJSON $outFile $inFile
2
```

3.9.2 Odhad počtu lidí žijící v budově

Data o počtu lidí žijících v budovách nikde přístupna nejsou. Existují záznamy o hustotách obyvatelstva pro katastrální území Prahy, které poskytuje Český statistický úřad[**?**]. Počet obyvatel lze přerozdělit mezi všechny budovy přes vážený průměr, kde váha je plocha budovy vypočítáné z polygonu ku celkové ploše všech budov v katastrálním území.

Pro budovy daného katastrálního území $B = \{b_1, b_2, ..., b_n\}$

a počet obyvatel katastrálního území K lze dostat pro budovu b_i odhad o_i obyvatel pro budovu

kde funkce $area : B \Longrightarrow \mathbb{R}$ získá plochu polygonu budovy

$o_i = \frac{area(b_i) * K}{\sum_i^n area(b_i)}$

avšak tímto se nezohlední více podlažní budovy, jelikož se použila čistě plocha půdorysu budovy. Existuje datová sada na GeoPortálu [7] ta dělí budovy uvedené v katastrálním úřadu na menší kusy, protože počet pater se může lišit podle části budovy. Data o částech budov postrádají(identifikátor)

informaci, ke kterému celku patří. Je potřeba části budovy přiřadit správně k původní budově. Nastává problém, že plocha budovy z katastrálního úřadu může obsahovat více části s informací o podlažích. Experimentoval jsem s několika možnostmi. Pro každou budovu sem vyhledal 8 nejbližších částí budov podle centroidů za pomoci akcelerační struktury(Ball tree, obdoba KDTree).

Zkoušel jsem následující testovací funkce na původní budovu b a část budovy v okolí $p \in \{p_1, ...p_8\}$ abych zjistil jestli fragment popisuje vybranou budovu. Tedy jsem hledal funkci s

- $s_1(p, b) = centroid(p) \in b$

  zde může nastat problém jelikož neplatí

  $centroid(p) \in p$ ukázáno v 3.3

- $s_2(p, b) = pointonsurface(p) \in p$

  pro tuto funkci platí,

  $pointonsurface(p) \in p$

- $s_3(p, b) = \frac{area(p \cap b)}{area(b)} > 0.95$

  Slovně fragment musí z alespoň 95% ležet uvnitř půdorysu budovy. Tato funkce má vyšší výpočetní náročnost.

Nejlepší výsledek poskytla funkce s_3 a tedy z 150,036 budov nalezla alespoň jeden fragment pro 117,539 budov. Následně když jsem měl mapování fragmentů budov zbývalo vypočítat celkovou plochu budovy, která bere v potaz podlaží.

Algoritmus pro výpočet plochy budovy s ohledem na podlaží b a nalezenými fragmenty $P = \{p_1, p_2, ..., p_i\}$ vypadal následovně, kde P mají funkci pro počet pater.

$numFloors : P \Longrightarrow \mathbb{N}$

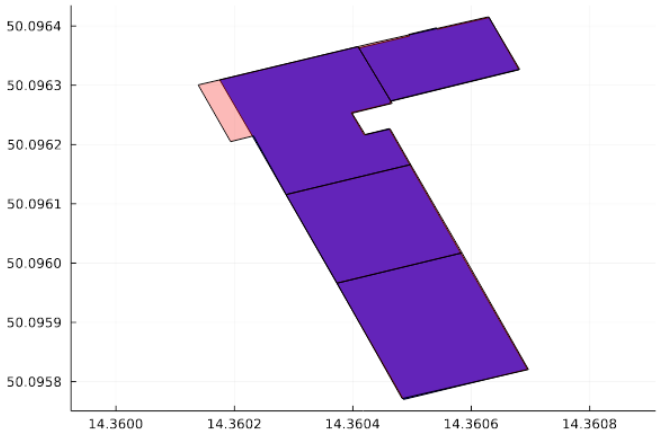
Pro zbylou část půdorysu pro kterou se nepodařilo nalézt fragment tak jsem zvolil že budou mít počet pater roven 3, což je průměrný počet pater z datové sady.

Výpočet obyvatel domu je tedy upraven nahrazením $area$ za $areaWrtFloors$ a tedy:

$o_i = \frac{areaWrtFloors(b_i) * K}{\sum_i^n areaWrtFloors(b_i)}$

Algorithm 1
Výpočet plochy budovy zohledňující počet pater uvedených v P

function AREAWRTFLOORS($b \in B$)
  $RemainingGeometryArea \leftarrow area(b)$
  $BuildingArea \leftarrow 0$
  for $t = 1 : i$ **do**
    $intersection_i \leftarrow p_i \cap RemainingGeometryArea$
    $RemainingGeometryArea \leftarrow RemainingGeometryArea \backslash intersection_i$
    $BuildingArea \leftarrow BuildingArea + numFloors(p_i) * area(p_i)$
  end for
  $BuildingArea \leftarrow BuildingArea + 3 * area(RemainingGeometryArea)$
  return $BuildingArea$
end function



Obrázek 3.2: Ukázka fragmentace budovy

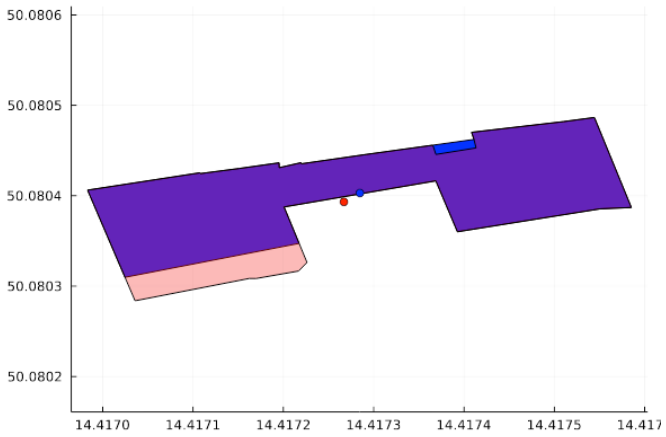
3.10 Práce s mapovými daty a hledání tras

Pro dohledání nejbližšího kontejneru na daný druh odpadu je třeba mapových data a možné práce s nimi. Hlavně vyhledávání pěší trasy mezi dvěma vybranými body. Je třeba najít $12 * 10^6$ tras.

$12 * 10^6$ = 150,000 (počet budov) * 10 (předvybraná nejbližší stanoviště) * 8 (počet druhu odpadu).

Obstaral jsem si mapová data pro Prahu z OpenStreet[8] map. Pro jejich načtení a práci s nimi jsem použil knihovnu(OpenStreetMapX.jl[51]). Tato knihovna umožňuje v mapě vyhledávání trasy mezi dvěma body.

Vyhledávání se uskutečnilo rozděleně podle druhu odpadu. Nemohl jsem ale spustit vyhledávání pro všechny budovy, jelikož polovina všech stanovišť na tříděný odpad je soukromá a tedy by se nemělo spustit vyhledávání pro všechny budovy ale pouze pro ty, které nemají soukromé kontejnery na



Obrázek 3.3: Ukázka, kdy centroid části budovy(červený bod) neleží uvnitř půdorysu budovy

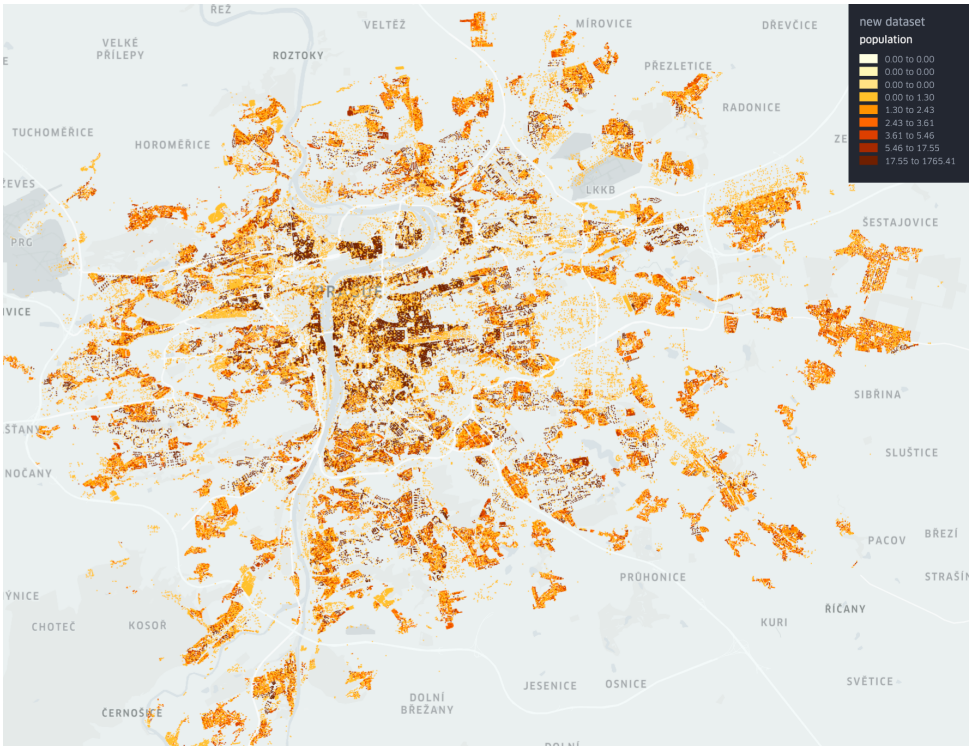
pozemku pro daný druh odpadu(plast, papír).

Pro získání budov bez soukromých stanovišt jsem vybral všechna soukromá stanoviště. Našel na které parcele se nachází. Ty označil a následně vyhledal všechny budovy, které leží na označených stanovištích. Tyto budovy jsem odebral ze seznamu všech budov, který sloužil pro vyhledávání nejbližších kontejnerů.

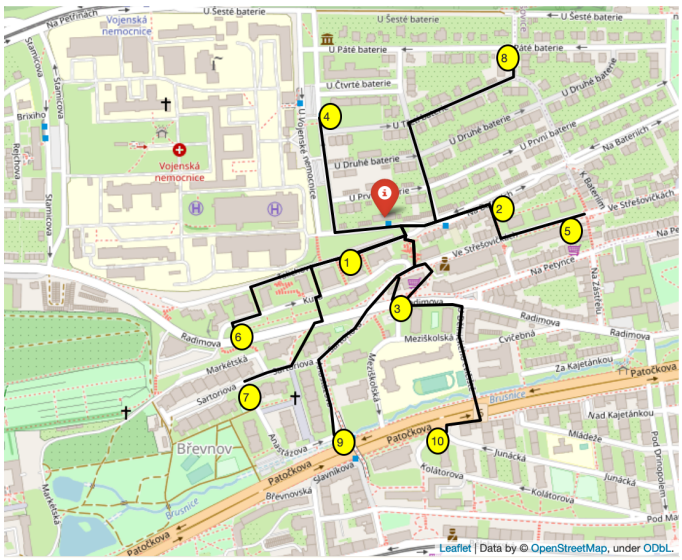
Samotné vyhledávání nejbližšího kontejneru pro daný druh odpadu vypadá následovně.

Algoritmus byl spuštěn ve více vláknech. Kde paralelizovanu částí byla funkce která k budově nalezne nejbližší kontejner(8 řádek algoritmu2).

Funkce *mappingExists* kontroluje, jestli již výpočet pro katastrální území proběhl a existuje soubor s mapováním na disku. Díky tomu lze běh algoritmu pozastavit a opětovně spustit bez ztráty výsledku.



Obrázek 3.4: Mapa Prahy s výslednou populací v budovách



]

Obrázek 3.5: nejbližší kontejnery

Algorithm 2
vyhledávání nejbližšího kontejneru pro každou budovu

```
1: function NEARESTCONTAINERFORBUILDINGS(trashType)
2:     locations ← getLocationsWithTrashType(trashType)
3:     for cadastralZone in cadastralZones do
4:         buildingsMapping ← {}(mapping between buildings and locations)
5:         if mappingExists(cadastralZone) then continue
6:         end if
7:         for b in cadastralZone.buildings do
8:             preselectedLocations ← knn(point(b), locations, 10)
9:             distances = []
10:            for i = 1 : 10 do
11:                distances[i] ← getGraphDistance(preselectedLocations[i], point(b))
12:            end for
13:            closestLocationId ← argmin(distances)
14:            buildingsMapping[b] ← preselectedLocations[closestLocationId]
15:        end for
16:        saveMapping(cadastralZone, buildingsMapping)
17:    end for
18: end function
```

3.11 Výsledek datové analýzy

Pro otestování, jestli je možné odhadovat rychlost plnění jsem použil data z měřených kontejnerů. Z měření jsem extrahoval faktor plnosti. Tedy poměr kdy byl kontejner prázdný a kdy plný. Použil jsem Pearsonovu korelaci pro zjištění, jestli se zde vyskytuje lineární vazba.

Druh odpadu	Počet stanovišť	Korelace s populací
Plast	119	0.345
Papír	121	0.208
Nápojové kartony	65	0.216
Kovy	24	0.012
Barevné sklo	64	0.056
Čiré sklo	62	0.281

Tabulka 3.1: Tabulka s výslednou Pearsonovou korelací

Vyskytují se zde slabé lineární korelace a může dávat smysl je použít pro určení rychlosti plnění kontejnerů. Nízkou korelaci u barevného skla by šlo vysvětlit tím že je zde překryv s čistým sklem a bylo by třeba upravit vyhledávání nebližšího kontejneru pro odpady typu sklo. A u kovů nízkým počtem dat. Faktor plnosti má několik nedostatků a použil

Částečné výsledky pro daná stanoviště jsem spojil s daty o stanovištích a uložil, aby mohly být ve webové aplikaci prezentovány.

Kapitola 4

Webová aplikace

V této kapitole popíši jak sem vytvořil webovou aplikac,i která umožní vyhledat kontejnery a jak dokáže prezentovat data ze současné datové, ale i budoucích analýzy. Zároveň i popíši jak vypadá architektura, jaké technologie byly použity, a také jak byla nasazena do produkčního prostředí pro veřejnou přístupnost.

4.1 Cíle webové aplikace

Cílemi webové aplikace je uživateli umožnit vyhledání kontejnerů v daného druhu odpadu v jeho okolí a prezentovat získanou současnou plnost. Webová aplikace má problém uživatele řešit rychle a efektivně a má být použitelná na mobilním zařízení a mít prvky PWA(více v 4.5). Zároveň by měla být vizuálně přívětivá aby motivovala uživatele k používání. A měla by existovat jak v české tak anglické verzi.

4.2 Prototyp

Vytvořil jsem prototyp webové aplikace, do kterého jsem zapracoval funkční požadavky. Díky tomu bylo snadné rychle iterovat s nápady. A ujasnit si vizuální podobu. Prototyp umožnujě efektivní implementaci a od návrhu vizuální podoby. Byl vytvářen ve webové službě Figma[9].

Barevná paleta se skládá z černé, bíle a ze stylizovaných barev kontejnerů, které tak jasně podtrhávájí k čemu je stránka určena. Vytvořil jsem i logo, které má v sobě tématiku tříděných kontejnerů a bude přítmno na všech stránkách.

(a) : Použitá paleta barev **(b) :** Logo webové stránky

Obrázek 4.1: Částí návrhu

4.3 Popis stránek

4.3.1 Hlavní stránka

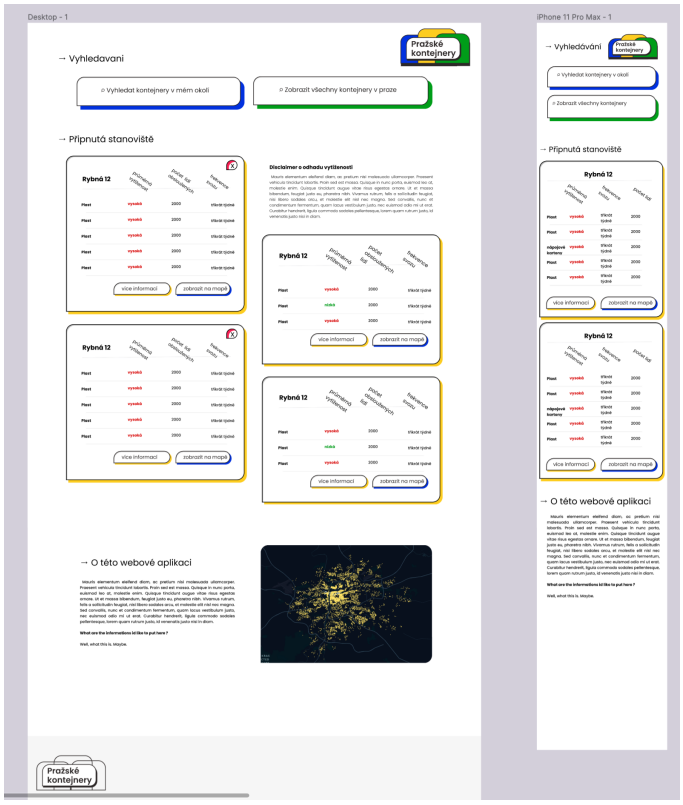
Na tuto stránku zavítá uživatel jako první. Je seznámen co může webová stránka dělat a v případě, že má připnutá stanoviště, tak se mu zde zobrazí. Nahoře se vyskytují dvě tlačítko. První dostane uživatele do formuláře s filtry pro vyhledávání stanovišť v okolí. Druhé zobrazí uživateli mapu všech stanovišť v Praze.

4.3.2 Filtry pro vyhledávání

Zde si uživatel vybere, které kontejnery v okolí chce vyhledat. Buď může zvolit, že chce vidět všechna stanoviště. A nebo zaškrtnutím nastaví filtr, že dané stanoviště musí obsahovat kontejnery, pro jím vybrané druhy odpadu. Zároveň je zde přítomen interaktivní obrázek s kontejnery, který se vybarvuje podle toho, jaké druhy odpadu uživatel vybral. Ten uživatele seznamuje s tím, jaká barva slouží pro jaký kontejner. Kliknutím na pokračovat se dostane na mapu s vyhledáváním stanovišť.

4.3.3 Vyhledávání stanovišť

Zde je již uživateli prezentována mapa Prahy. Zobrazí se požadavek na polohu a po potvrzení se mapa přiblíží na uživatelovu reálnou polohu a zobrazí se jím specifikované kontejnery v okolí. Na mapových značkách pro stanoviště jsou zobrazeny kontejnery pro druhy odpadů, které se na stanovišti



Obrázek 4.2: Návrh hlavní stránky, jsou přítomny jak návrh pro počítačovou verzi tak pro mobilní

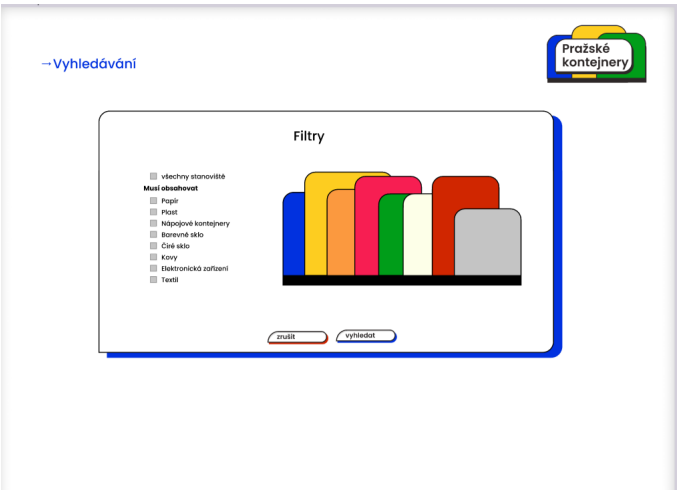
skutečně nachází. Kliknutím na značku se uživateli zobrazí okno s bližšími detaily stanoviště. Je zde tabulka, kde každý řádek reprezentuje jeden druh odpadu, pro který se na stanovišti nachází kontejner. V tabulce lze prezentovat jak reálné informace, tak výsledky datové analýzy. Kliknutím na "zobrazit detaily"se dostane na stránku s více detaily pro stanoviště.

4.3.4 Detaily stanoviště

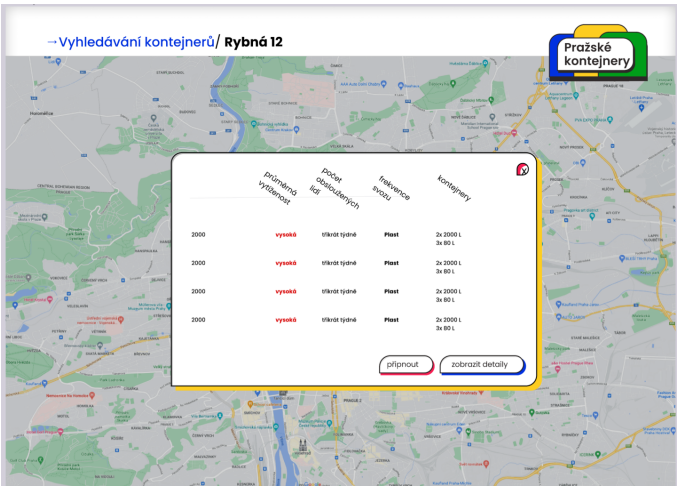
Tato stránka je určena pro detailní zobrazení stanoviště. Zde uživatel nalezne stejné informac,e jako při rozkliknutí vyhledávání stanovišť4.3.3, je zde prostor pro více informaci. URL daného stanoviště lze použít a sdílet s ostatními. Zároveň se zde nachází text, který uživatle seznamuje s tím, jak predikce plnosti funguje, a že se nejedná o reálnou ale čistě vypočítanou informaci.

4.3.5 Všechna stanoviště

Stránka je podobná vyhledávání stanovišť, ale uživatel zde nalezne všechna stanoviště s tříděným odpadem v Praze. Stanoviště může filtrovat podobně



Obrázek 4.3: Stránka s filtry pro hledání kontejnerů v okolí

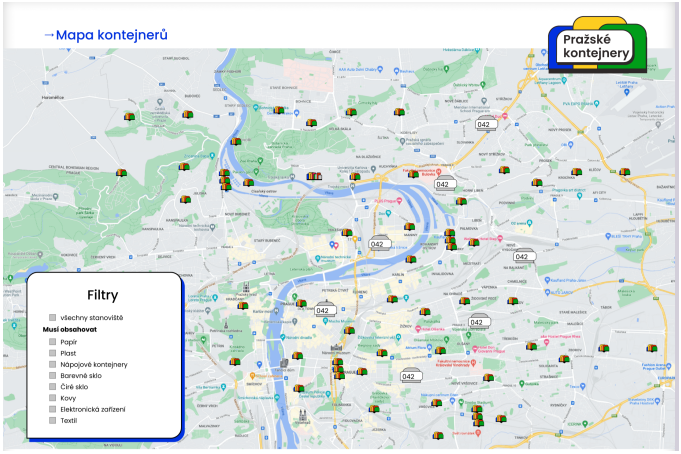


Obrázek 4.4: Okno po rozkliknutí značky stanoviště.

jako v 4.3.2. Změna filtru se ihned aplikuje.

4.4 Implementace webové části

Webovou stránku jsem implementoval ve webovém frameworku Next.JS[10], který rozšiřuje React[11] o možnosti statické generace stránky(static site generation - SSG) a serverové renderování(server side rendering - SSR). Takže má programátor k dispozici jak SPA, tak i SSR a SSG a je na jeho uvážení jaký způsob využije na základě stránky kterou implementuje. Výhodou SSR, SSG může být rychlejší prvotní načítací doba a možnost, aby stránka mohlá být lépe indexována webovými vyhledávači a tím pádem by na něm mohla být dohledatelná.



Obrázek 4.5: Stránka s vyhledáváním kontejnerů

4.4.1 Použité technologie

- **Typescript**([28] - Nadstavba nad jazykem JavaScript, která ho rozšiřuje o statické typování. Kompiluje se do JavaScriptu.
- **React**[11] - Knihovna pro tvorbu webových uživatelských rozhraní.
- **NextJS**[10] - Framework využívající React a umožňuje jeho použití na serveru.
- **MapboxGL**[12] - Slouží pro zobrazování map a umožňuje geospaciální vizualizace.
- **Tailwind CSS**[13]- Knihovna pro efektivnější stylování v CSS.
- **i18next**[14] - (není nijak asociována s NextJS) Knihovna umožňující snadné psaní stránky za podpory více jazyků v uživatelském rozhraní.
- **Axios**[15] - HTTP klient.
- **React Query**[16] - Knihovna pro synchronizaci dat z backendu a snadné používání v Reactu, vytváří cache vrstvu a umožnuje držet data v aktuálnosti.

4.4.2 Mapy

Důležitou částí webové stránky jsou mapy, kde jsou kontejnery vizualizovány. Pro to jsem použil již zmíněnou službu MapboxGL[12]. Ta umožňujě snadné zobrazování map a poskytuje rozhraní pro vizualizaci vlastních GeoJSON dat.

4.5 Prvky progresivní webové aplikace

Stránka má obsahovt prvky PWA(progresivní webová aplikace). Takové stránky se přibližují funkcionalitou mobilním aplikacím.

Jsou implementovány následující prvky ze seznamu.

- **Instalovatelnost** - Stránka lze nainstalovat, a tím se přidá na do seznamu aplikací na uživatelově zařízení.
- **Responzivita** - Rozložení stránky se přizpůsobuje velikosti obrazovky uživatelova zařízení
- **Persistence uživatelských dat** - Uživatel může připnout stanoviště, ta se zobrazí na hlavní stránce. Stránka si tato připnutá stanoviště pamatuje a při opětovném otevření je uživateli zobrazí
- **Dohledatelnost** - Stránka splňuje požadavaky aby mohla být zaindexována webovými vyhledávači.

Obrázek 4.6: Ukázka nainstalované aplikace

4.6 Server

Server poskytuje REST API rozhraní pro webovou stránku. Obsluhuje požadavky pro vyhledávání nejbližších stanovišť, detaily o konkrétním stanovišti a požadavek pro všechny stanoviště. Server při startu načte GeoJSON soubor se všemy daty o stanovištích a podle požadavku data z tohoto souboru filtruje. Data odešle uživateli opět ve formátu GeoJSON.

Pro vyhledávání nejbližších kontejnerů je využita akcelerační strukturu BallTree, která je specifická pro filtry druhu odpadu a dostupnosti, ta se po vytvoření zacachuje a následující požadavky se stejnými filtry jsou obstarány rychleji. Maximální kapacita cache je 10 a používá LRU(least recently used) pravidla pro nahrazování, když je cache plná.

Popis	**URI**	**Query parametry**
Získání 10 nejbližších stanovišť v okolí	`/api/v1/nearest-locations`	**lat**: zeměpisná šírka uživatele **lng**: zeměpisná výška uživatele **trashTypes**: pole identifikátorů odpadu **onlyPublic**: přepínač pro zobrazení pouze veřejných stanovišť
Získání všech stanovišť v okolí	`/api/v1/all-locations`	
Získá stanoviště podle pole identifikátorů	`/api/v1/locations`	**locationIds**: pole identifikátorů stanovišť

Tabulka 4.1: Endpointy serveru

4.7 Nasazení do produkce

Pro nasazení webové aplikace jsem použil Docker[29]. Díky virtualizaci umožňuje snadné zabalení aplikace, nasazení a aktualizaci. Jelikož se pokaždé začíná s čistým operačním systémem nenastavá zde kumulace skrytého stavu, který by později mohl způsobovat chyby. Lze využít předpřipravené kontejnery, v kterých již nainstalováno prostředí pro vybraný jazyk. Kontejnery se konfigurují pomocí textový souborů dockerfile.

Nasazená stránka se skládá z následujících kontejnerů:

- **Server** - Vychází z kontejneru [32]
- **Webová stránka** - Vychází z kontejneru [35] a používá modifikovaný dockerfile z [37]
- **Proxy** - Používá kontejner [33], v kterém je program Nginx([38]), ten sjednocuje rozhraní serveru a frontendu. Aplikace tak může běžet na sjednocené URL.

  Požadavky na **https://prazskekontejnery.cz/api/v1/** proxy pošle na kontejner se serverem Zbytek požadavků proxy pošle na kontejner s frontendem

- **Certbot** - Používá kontejner[34], v kterém je program Certbot [39], který slouží k obstarávání certifikátů pro HTTPS.

Pro zpřístupnění stránky pro veřejnost jsem pořídil doménu **prazskekontejnery.cz** na službě Active24[40]). Pro hosting jsem využil VPS(virtual private server) od služby DigitalOcean[41]. Kvuli možným vyšším nárokům na paměť kvůli runtime jazyka Julia a cache vyhledávácích stromů jsem zvolil variantu s 1 sdíleným CPU a 2GB RAM.

Kapitola 5

Uživatelské testování

Webovou aplikaci jsem podrobil kvalitativnímu uživatelskému testování. Zvolil jsem metodu Quick and dirty[2]. To je metoda, která umožňuje rychle získat zpětnou vazbu od uživatelů a je vhodná pro rychlé iterace. Zadavtel prezentuje uživateli stránku a ptá se, jak si uživatel myslí že by mohl dosáhnout zadaného cíle. Pozoruje, jak se stránkou interagují a ptá se, jak interpetují co je jim zobrazováno.

Vybral jsem dva scénáře na testování:

- nalézt nejbližší stanoviště, kam by mohli vytříd kovy
- vyhledat kontejner s elektrozařízením v okolí uživatelova bydliště

Aplikaci jsem testoval na 5 uživatelích ve věkovém rozmezí 20-25.

Uživatelům sem prezentoval tyto dva problémy spolu s URL webové stránky, kterou si měli otevřít na svém mobilním zařízení. Bez mého zásahu jsem pozoroval, jak stránku používájí k řešení.

Uživatelé nalezli nejbližší stanoviště s kontejnerem na kov do 2 minut.

U jednoho uživatele nastal problém s tím, že zařízení odmítlo zjistit jeho polohu. Chyba nebyla na stránce nijak prezentována a uživatel musel použít druhou stránku, která zobrazí všechna stanoviště, a zde na mapě ručně vyhledal svou polohu. Jednomu uživateli přišlo méně přehledné zobrazení značek se stanovišti na mapě. Dvoum uživatelům se zobrazila mapa a po zjištění polohy jim mapa nebyla přiblížena.

U druhého problému jsem u většiny uživatelů pozoroval chvilkové zmatení ve stavu po kliknutí na "Zobrazit všechna stanoviště". Zobrazila se jim mapa bez stanovišť, která se načítala na pozadí. Došlo k chvilkovému zmatení kterému by bylo dobré předejít.

Uživatelům nebyla jasná funkcionalita filtrů. Očekávali, že se jim zobrazí stanoviště, která obsahují alespoň jeden druh, jimi označeného, odpadu. Avšak zobrazila se pouze ta stanoviště, která obsahovala všechny, jimi označené, druhy odpadu.

Kapitola 6

Závěr

Cílem bakalářské práce bylo vytvořit webovou aplikaci, na které může uživatel vyhledat stanoviště s kontejnery na tříděný odpad a zároveň mu je prezentován odhadovaný současný stav zaplnění.

Pro predikci plnosti jsem zkoumal Pražská data a zjišťoval pomocí geospaciální datové analýzý, jestli lze odhadnout rychlost plnění jednotlivých kontejnerů. K tomu jsem využil myšlenky voronoi diagramu a pomocí ní zjišťoval pro každý kontejner, kolik na něj spadá lidí. Získané počty lidí jsem porovnával s vytížeností měřených kontejnerů. Nalezl jsem slabé korelace, které ukázaly, že by bylo možné tuto metodu rozvinout a získané hodnoty použít pro výpočet rychlosti plnění. Pro možnost odhadovat současnou plnost kontejnerů je třeba vyčkat na data o svozech odpadu od projektu "RFID odpadové nádoby"2.1.2.

Vytvořená webová aplikace uživatelům dává možnost vyhledat stanoviště na tříděný odpad, buď přes polohu uživatelova zařízení, kde mu jsou zobrazeny pouze nejbližší kontejnery v okolí, a nebo manuálně přes mapu všech kontejnerů v Praze. A zatím prezentuje uživateli jen hrubé výsledky analýzy.

6.1 Další kroky

Jsou dvě oblasti kterými je možné navázat na tuto práci. První částí je samotná predikce a druhou je webová aplikace.

Pro lepší odhad, kolik lidí obsluhuje jaký kontejner, by byl vytvořen dotazník, jehož cílem by bylo zjistit, jak lidé třídí odpad a jak se rozhodují při výběru stanoviště. Získáne výsledky by byly zapracovány do modelu odhadu. Dále je třeba vyřešit problém s aktuálností dat. Přidat službu, která v pravidelných intervalech aktualizuje data a provádí pro ně nové výpočty rychlosti plnění.

Do webové aplikace by byly zapracovány zpozorované nedostatky z kvalitativního testování 5. Byla by zlepšena komunikace s uživatelem o to co se děje v aplikaci. Zejména při neůspěšném zjištění polohy a u požadavků na server, které trvají delší dobu. Dále by mělo být zlepšeno uživatelské rozhraní pro mobilní zařízení. A pro monitorování návštěvnosti by byly zavedeny webové analytiky.

Literatura

[1] Tomáš Werner, *Optimalizace*, Katedra kybernetiky, Fakulta elektrotechnická, České vysoké učení technické, 2022

[2] Leah Buley, *The user experience team of one, A Research and Design Survival Guide*, Rosenfeld Media 2013

[3] Operátor ICT a.s. (2022) *Golemio* [online]
`https://golemio.cz`

[4] Smart Prague (2022) [online]
`https://www.smartprague.eu`

[5] Operátor ICT a.s. (2022)
`https://operatorict.cz/`

[6] Státní správa zeměměřictví a katastru (2022) *Katastrální mapa ČR ve formátu SHP distribuovaná po katastrálních územích (KM-KU-SHP)* [data]
`https://services.cuzk.cz/shp/ku/epsg-5514/`

[7] Institut plánování a rozvoje hl. m. Prahy (2022) *Podlažnosti - CZ-70883858-URK_CUR.URK_SS_Podlaznost_p* - (CC BY-SA 4.0) [online]
`https://www.geoportalpraha.cz/cs/data/metadata/601B7F51-EBA2-4E86-9B02-10AE47A1FF19`

[8] OpenStreetMap (2022) [online]
`https://www.openstreetmap.org`

[9] Figma, Inc *Figma* [online]
`https://figma.com/`

[10] Vercel, Inc (2022) *Next.js* [online]
`https://nextjs.org/`

[11] Meta Platforms, Inc (2022) *React* [online]
https://reactjs.org/

[12] *Mapbox* (2022) [online]
https://www.mapbox.com/

[13] Tailwind Labs Inc (2022) *Tailwind css* [online]
https://tailwindcss.com/

[14] i18next (2022) *i18next* [online]
https://www.i18next.com/

[15] Axios (2022) *Axios* [online]
https://axios-http.com/

[16] Tanner Linsley (2022) React Query [online]
https://react-query.tanstack.com/

[17] Opendata hlavního města Prahy (2022)
https://opendata.praha.eu/

[18] https://data.gov.cz/ Ministerstvo vnitra České republiky (2022) *Portál otevřených dat* [online]

[19] https://www.esfcr.cz/projekty-opz/-/
asset_publisher/ODuZumtPTtTa/content/
implementace-strategii-v-oblasti-otevrenych-dat-ii

[20] Operátor ICT a.s. (2022) *Obecná dokumentace Datové platformy* [online]
https://operator-ict.gitlab.io/golemio/documentation/

[21] Smart Prague (2022) *Chytrý svoz odpadu* [online]
https://www.smartprague.eu/projekty/chytry-svoz-odpadu

[22] Smart Prague (2022), *RFID odpadové nádoby* [online]
https://www.smartprague.eu/projekty/rfid-odpadove-nadoby

[23] Institut plánování a rozvoje hl. m. Prahy (2022), *Pražská integrovaná doprava - vstupy do metra* - (CC BY-SA 4.0) [online]
https://www.geoportalpraha.cz/cs/data/otevrena-data/
8A9EB9A8-FEBD-4A24-972F-48182C1A6D62

[24] Data Carpentry (2022), *Introduction to Raster Data* - (CC-BY 4.0) [online]
https://datacarpentry.org/organization-geospatial/
01-intro-raster-data/

[25] Český statistický úřad (2021) *Obyvatelstvo a rozloha katastrálních území Prahy 2000-2020* https://www.czso.cz/csu/xa/dalsi_casove_rady_
obyvatelstvo

[26] Operátor ICT, a.s. (MIT) [online]
https://gitlab.com/operator-ict/golemio/code/modules/sorted-waste-stations/-/blob/development/src/output-gateway/sorted-waste-stations-pg/models/SortedWasteStationsModel.ts

[27] Weisstein, Eric W, MathWorld–A Wolfram Web Resource, *Voronoi Diagram.* https://mathworld.wolfram.com/VoronoiDiagram.html *Voronoi Diagram* [online]
https://mathworld.wolfram.com/VoronoiDiagram.html

[28] Microsoft Corporation (2022) *TypeScript* [online]
https://www.typescriptlang.org/

[29] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.
https://www.docker.com/

[30] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and contributors (2022) *The Julia Programming Language* [online]
https://julialang.org/

[31] Yeesian Ng, Maarten Pronk, Martijn Visser, and contributors *ArchGDAL.jl* [online]
https://yeesian.com/ArchGDAL.jl/latest/

[32] https://hub.docker.com/_/julia

[33] https://hub.docker.com/_/nginx

[34] https://hub.docker.com/r/certbot/certbot

[35] https://hub.docker.com/_/node

[36] https://docs.docker.com/engine/reference/builder/

[37] Vercel, Inc. (2022) *Dockerfile* [online]
https://github.com/vercel/next.js/tree/canary/examples/with-docker

[38] F5, Inc. (2022) *Nginx* [online]
https://www.nginx.com/

[39] Electronic Frontier Foundation (EFF) (2022) [online]
https://certbot.eff.org/

[40] ACTIVE 24, s.r.o. (2022) [online]
https://www.active24.cz/

[41] DigitalOcean, LLC (2022) *DigitalOcean – The developer cloud* [online]
https://www.digitalocean.com/

[42] Esri (2022) *Shapefiles* [online]
https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm

[43] Internet Engineering Task Force (IETF) *The GeoJSON Specification (RFC 7946)* [online]
https://geojson.org/

[44] *Literate programing* (2022) [online]
http://www.literateprogramming.com/

[45] *Project Jupyter* (2022) [online]
https://jupyter.org/

[46] *Kepler.gl* [online]
https://kepler.gl/

[47] Jakub Zíka (2022) *Unused query parameter accesibility* [online]
https://gitlab.com/operator-ict/golemio/code/modules/sorted-waste-stations/-/issues/13

[48] *NGA Standardization Document*, Department of Defense, World Geodetic System 1984 (updated 8 July 2014) [online]
https://earth-info.nga.mil/index.php?dir=wgs84&action=wgs84

[49] Doc. Ing. Václav Čada CSc., Západočeská univerzita, Fakulta aplikovaných věd, Katedra matematiky *Souřadnicové systémy* (2013) [online]
https://web.archive.org/web/20130517195525/http://gis.zcu.cz/studium/gen1/html/ch02s03.html

[50] Jan Šimbera, Přírodovědecká fakulta Univerzity Karlovy v Praze, Souřadnicové systémy (2018) [online]
https://www.natur.cuni.cz/geografie/geoinformatika-kartografie/ke-stazeni/projekty/moderni-geoinformacni-metody-ve-vyuce-gis-kartografie-a-dpz/souradnicove-systemy/

[51] Przemysław Szufel and contributors (2022) *OpenStreetMapX.jl* [online]
https://pszufe.github.io/OpenStreetMapX.jl/stable/

Příloha A

Struktura přiložených souborů

- **Explorations/** - kód datové analýzy
- **be/** - kód serveru
- **fe/** - kód webové stránky
- **nginx/** - konfigurace proxy