Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19 Bratislava

Webová aplikácia s POSTGIS databázou a Mapbox API

Semestrálny projekt

Martin Gašpar

Študijný odbor: Inteligentné softvérové systémy
Ročník: 2.
Predmet: Pokročilé databázové technológie
Cvičiaci: Ing. Miroslav Rác
Akademický rok: 2018/2019

Návrh zadania

Cieľom webovej aplikácie je zobrazenie nehodovosti chodcov a cyklistov v rámci vybraného mesta, vplyv svetelnej signalizácie na počet nehôd, farebná mapy lokalít podľa počtu nehôd v okolí a vplyv vzdialenosti nemocnice na úmrtnosť. Možnosť filtrovania dát poskytuje ďalšie možnosti vyhodnocovania týchto dát ako je napr. vplyv počasia.

Databáza

Databázovy server

Pre uchovávanie dát bola zvolená databáza PostgreSQL vo verzii 10.5 rozšírená o PostGIS vo verzii 2.5.0 .
Kvôli importu dát bola vytvorená nová DB aj s heslom „heslo“.

Dataset a spracovanie

Ako dataset som si vybral dáta o nehodovosti cyklistov a chodcov v meste Chapel Hill, v Severnej Karolíne. Získané boli zo stránky https://www.data.gov. Táto stránka obsahuje veľké množstvo datasetov s dátami z USA. Neskôr boli doplnené o datasety obsahujúce Neighborhoods a polôh svetelnej signalizácie.
Import dát do DB bol vykonaní pomocou ogr2ogr funkcie, ktorá je súčasťou Geospatial Data Abstraction Library (GDAL). Tá bola inštalovaná cez brew vo verzii 2.3.1. Príklad pre dáta o svetelnej signalizácii.

```
ogr2ogr -f "PostgreSQL" PG:"dbname=test user=postgres password=heslo"
"path/to/file/traffic-signals-in-chapel-hill.geojson" -skip-failures
```

Dáta o nehodovosti sme spojili do jednej veľkej tabuľky crash_data_chapel_hill_region s niektorými používanými a spoločnými informáciami napr. o počasí alebo zranení. Ešte predtým sme však obe tabuľky rozšírili o stĺpec crashtype, aby me ich po spojení vedeli rozlíšiť.

```
ALTER TABLE bicycle_crash_data_chapel_hill_region ADD COLUMN crashtype bicycle;

ALTER TABLE pedestrian_crashes_chapel_hill_region ADD COLUMN crashtype pedestrian;


insert into crash_data_chapel_hill_region (weather, injur, crash_mont, geo_point_2d, wkb_geometry, crash_type) (select

weather, bike_injur as injur, crash_mont, geo_point_2d, wkb_geometry, crashtype as crash_type from

bicycle_crash_data_chapel_hill_region);


insert into crash_data_chapel_hill_region(weather, injur, crash_mont, geo_point_2d, wkb_geometry, crash_type) (select weather,

ped_injury as injur, crash_mont, geo_point_2d, wkb_geometry, crashtype as crash_type from

pedestrian_crashes_chapel_hill_region);
```

Dáta s polohami svetelných signalizácii nevyžadovali žiadnu úpravu, naopak dáta Neighborhoods boli pre vybraný prípad použitia doplnené a počet nehôd v definovanej vzdialenosti od nich.

```
do
$$
declare
 i record;
begin
 for i in 1..246 loop
  IF (SELECT COUNT(DISTINCT bc.wkb_geometry) as hodnota FROM (SELECT * FROM neighborhoods WHERE ogc_fid = i)
as x JOIN crash_data_chapel_hill_region bc ON ST_DistanceSphere(x.wkb_geometry, bc.wkb_geometry) <= 50 GROUP BY
x.wkb_geometry) > 0 THEN
  update neighborhoods set accident_number=
  (SELECT COUNT(DISTINCT bc.wkb_geometry) as hodnota FROM (SELECT * FROM neighborhoods WHERE ogc_fid = i)
as x JOIN crash_data_chapel_hill_region bc ON ST_DistanceSphere(x.wkb_geometry, bc.wkb_geometry) <= 50 GROUP BY
x.wkb_geometry) WHERE neighborhoods.ogc_fid = i ;
 END IF;
end loop;
end;
$$
```

Upravený algoritmus pre tabuľku chapel_hill_bike_map_lines. Pre jeho použitie sme museli zmazať a nanovo vytvoriť primary key.

```
do
$$
declare
 i record;
begin
 for i in 1..123 loop
  IF (SELECT COUNT(DISTINCT bc.wkb_geometry) as hodnota FROM (SELECT * FROM chapel_hill_bike_map_lines
WHERE id = i) as x JOIN crash_data_chapel_hill_region bc ON ST_DistanceSphere(x.wkb_geometry, bc.wkb_geometry) <= 50
GROUP BY x.wkb_geometry) > 0 THEN
  update chapel_hill_bike_map_lines set accident_number=
  (SELECT COUNT(DISTINCT bc.wkb_geometry) as hodnota FROM (SELECT * FROM chapel_hill_bike_map_lines WHERE
id = i) as x JOIN crash_data_chapel_hill_region bc ON ST_DistanceSphere(x.wkb_geometry, bc.wkb_geometry) <= 50 GROUP
BY x.wkb_geometry) WHERE chapel_hill_bike_map_lines.id = i ;
 END IF;
end loop;
```

```
end;
$$
```

Aplikácia

Frontend

Frontend aplikácie je postavený na troch jazykoch HTML, CSS a Javascript. Funkcionalita aplikácie vrátane integrácie Mapbox API je realizovaná cez Javascript.
Menu s funkcionalitou aplikácie sa zobrazí sa po kliknutí na t

Backend

Ako jazyk pre backend bol vybraný jazyk PHP pre svoju jednoduchosť, stále veľkú obľubu a predchádzajúce skúsenosti s týmto jazykom.
Výsledok query v podobe GEOJSONu je následne upravený do správnej podoby a ako JSON odoslaný na zobrazenie pomocou Mapbox API.

Webserver

Ako webserver bol zvolený Apache webserver resp. neskôr bol používaný obyčajný lokalhost pre PHP server. Spúšťaní bol na počítači s operačným systémom macOS.
Spúšťaní bol cez príkaz:

```
sudo /usr/bin/php -S localhost:80
```

Služby

Vychádzajúc zo získaných a spracovaných dát, aplikácia ponúka nižšie uvedené služby. Potrebné dáta sú vracané vo formáte GeoJSON a preto je ich možné bez ďalšej úpravy použiť na vykreslenie cez Mapbox API.

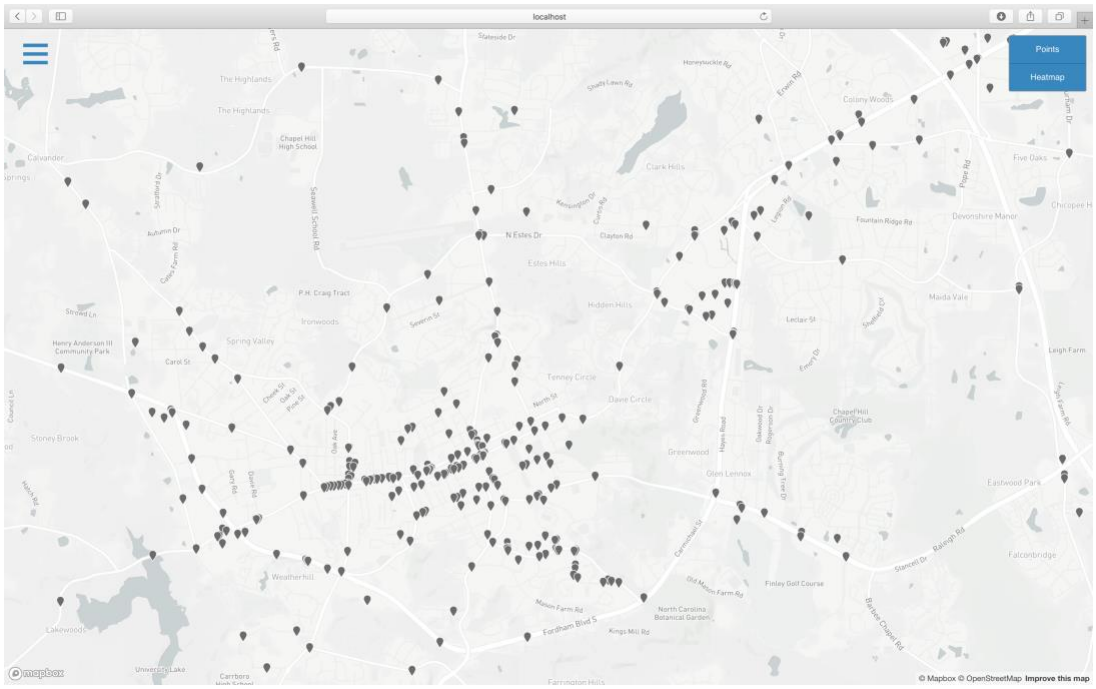
Zobrazenie dát a možnosť filtrovania – function1.php

Nasledujúca služba vráti všetky nehody v rámci mesta Chapel Hill. K dispozícii je niekoľko filtrov napríklad podľa typu zranenenia, počasia alebo mesiaca, kedy sa nehoda stala. Tiež je možné výsledky zobraziť vo forme tzv. markerov alebo ako heatmapu, z ktorej je možné zistiť lokality najčastejších nehôd.
Výsledky vraciame vo formáte FeatureCollection, ktorý obsahuje pole jednotlivých feature. Takíto tvar dát je používaný vo viacerých príkladoch pre Mapbox

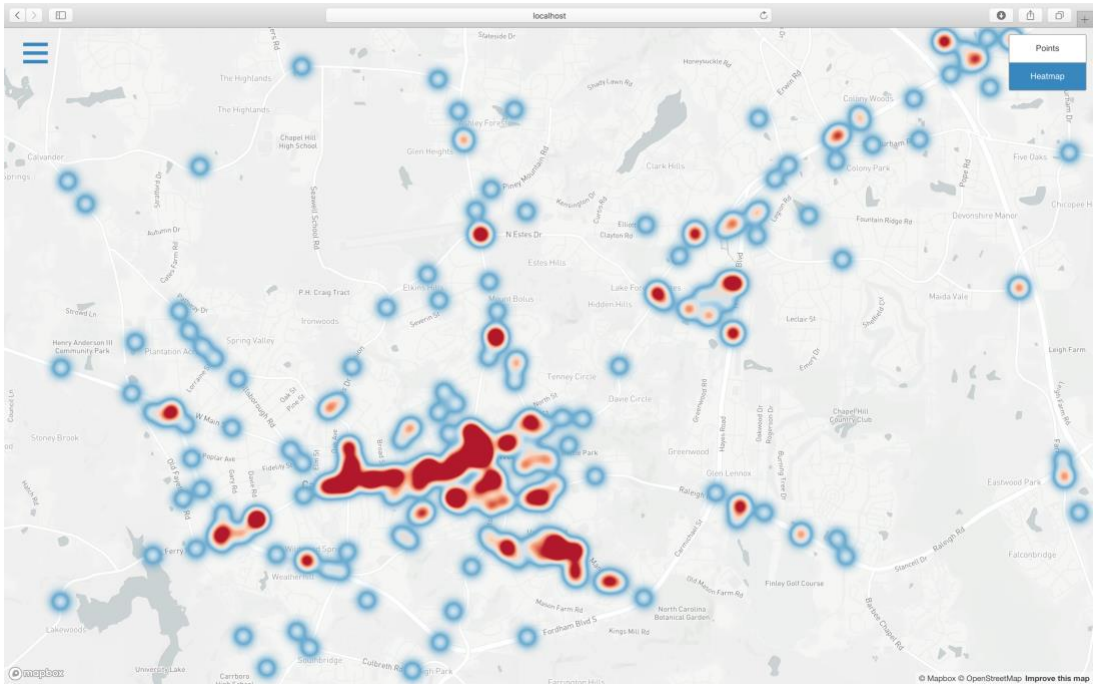
Query

```
SELECT jsonb_build_object('type','FeatureCollection','features', jsonb_agg(features.feature))FROM (SELECT
jsonb_build_object('type','Feature','geometry',ST_AsGeoJSON(ST_Transform(wkb_geometry, 4326))) AS feature FROM
(SELECT * FROM crash_data_chapel_hill_region $where) inputs) features;
```

Zobrazenie



Obrázok 1: Zobrazenie dát o nehodovosti vo forme bodov



Obrázok 2: Zobrazenie dát o nehodovosti vo forme heatmapy

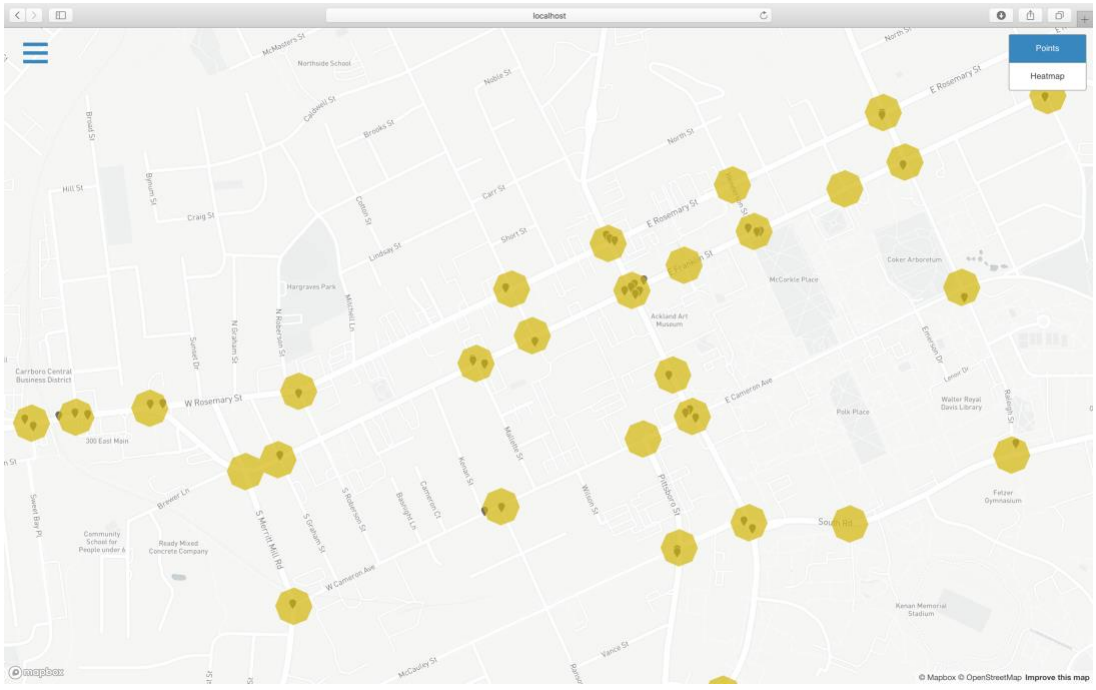
Zobrazenie svetelnej signalizácie a nehôd do zvolenej vzdialenosti – function3.php + function4.php

Predstavuje iný typ filtra, kedy sú vracané dáta o nehodách, ktoré sa stali vo vzdialenosti do 30m od svetelnej signalizácie. Výsledkom je polygón ohraničujúci vzdialenosť, ktorá používame v selecte pre získanie daných nehôd.
Využívame funkciu WITH RECURSIVE, kde si pred pripravíme polygóny a tie následne používame vo funkcii ST_Within pre určenie, či bod je alebo nie je v rámci hraníc polygónu.

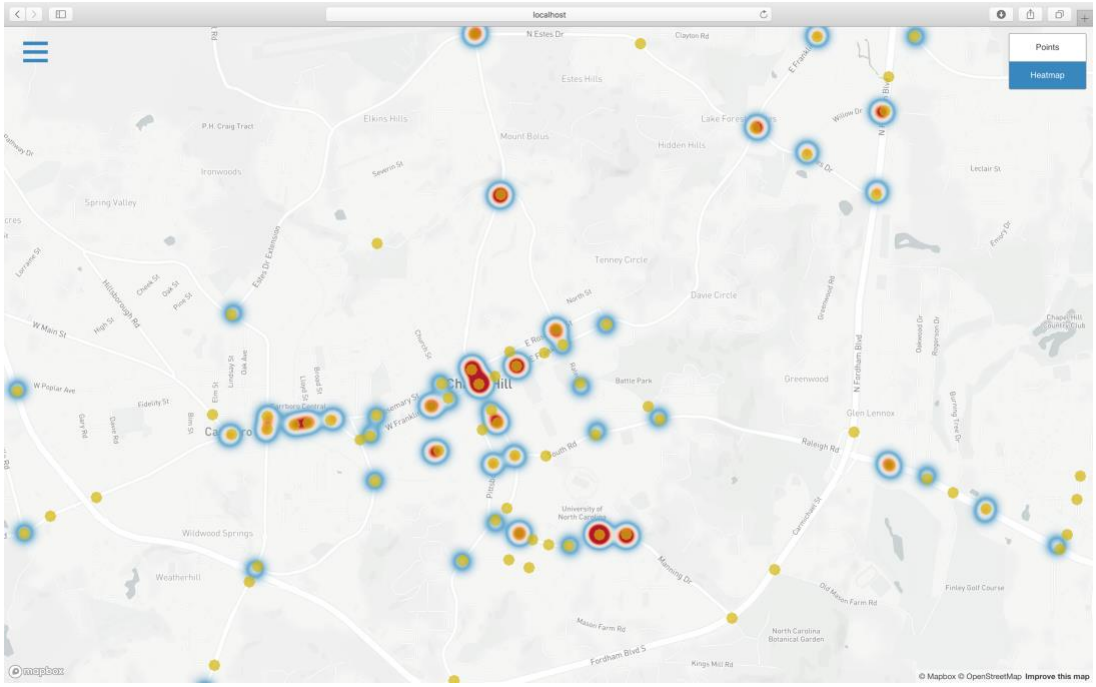
Query

```
WITH RECURSIVE t(n, geom) AS (
   SELECT 1, (SELECT ST_Buffer(CAST(ST_SetSRID(wkb_geometry, 4326) AS geography), 30, 'quad_segs=2') FROM
traffic_signals_in_chapel_hill WHERE ogc_fid = 1)
   UNION ALL
   SELECT n+1, (SELECT ST_Buffer(CAST(ST_SetSRID(wkb_geometry, 4326) AS geography), 30, 'quad_segs=2') FROM
traffic_signals_in_chapel_hill WHERE ogc_fid = n+1 AND n < (SELECT COUNT(ogc_fid) FROM traffic_signals_in_chapel_hill))
FROM t
)
SELECT jsonb_build_object('type','FeatureCollection','features', jsonb_agg(features.feature))FROM (SELECT
jsonb_build_object('type','Feature','geometry',ST_AsGeoJSON(ST_Transform(wkb_geometry, 4326))) AS feature FROM
((SELECT * FROM t LIMIT 122) as pol JOIN bicycle_crash_data_chapel_hill_region bc ON ST_Within(bc.wkb_geometry,
CAST(pol.geom AS geometry))) inputs) features;
```

Zobrazenie



Obrázok 3: Zobrazenie nehôd pri svetelnej signalizácii vo forme bodov



Obrázok 4: Zobrazenie nehôd pri svetelnej signalizácii vo forme heatmapy

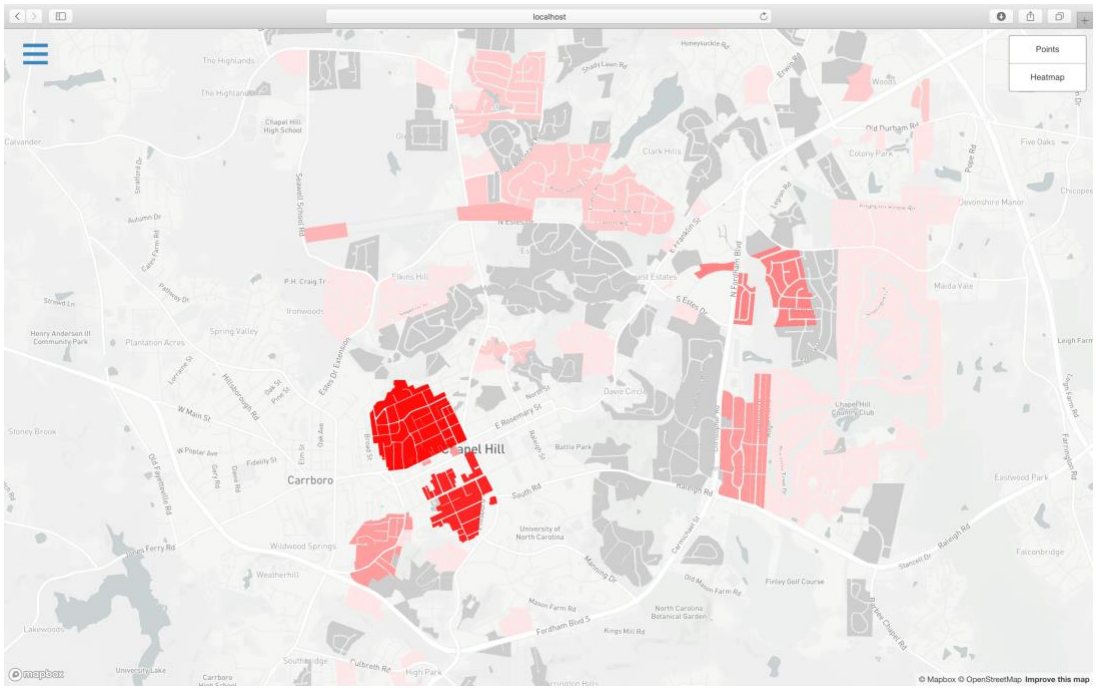
Zobrazenie polygónov lokalít s ofarbením na základe ich počtu – function8.php

Pri tomto prípade použitia sme museli vykonať predspracovanie, kde sme ku lokalitám do tabuľky pridali aj počet nehôd. Je to z dôvodu, že sme nenašli vhodný spôsob realtime vykonávaného query, ktoré by nebežalo nad 10s. Indexy sme nevedeli použiť z dôvodu, že vyžadujú viac miesta ako index umožňuje, *Index row requires 10728 bytes maximum size is 8191*.

Query

```
SELECT jsonb_build_object('type','FeatureCollection','features', jsonb_agg(features.feature)) FROM ( SELECT
jsonb_build_object('type','Feature','geometry',ST_AsGeoJSON(ST_AsText(wkb_geometry))::jsonb, 'properties', to_jsonb(inputs)-
'alt_city' - 'objectid' - 'org_name' - 'org_type' - 'alt_email' - 'alt_fname' - 'alt_lname' - 'alt_phone' - 'alt_state' - 'geo_shape' -
'prim_city' - 'prim_fname' - 'prim_email' - 'prim_lname' - 'prim_phone' - 'prim_state' - 'contactdate'
- 'alt_web_site' - 'alt_zip_code' - 'geo_point_2d' - 'prim_address' - 'wkb_geometry' - 'shape_starea'- 'ogc_fid' - 'prim_website' -
'prim_zip_code' - 'org_type_other' - 'shape_stlength' - 'alt_st_address' ) AS feature FROM (SELECT * FROM neighborhoods
LIMIT 246) inputs) features;
```

Zobrazenie



Obrázok 5: Zobrazenie lokalít s ofarbením na základe počtu nehôd

Zobrazenie polygónov pre vzdialenosť od nemocnice – function6.php

Pre tento účel sme pridali do DB tabuľka, ktorá obsahovala jediný záznam a to s polohou nemocnice. Chceli sme tak umožniť pomocou filtra analýzu nehôd napr. vplyv vzdialenosti nehody od nemocnice na úmrtnosť.