1. Prehľad

Webová aplikácia na zobrazenie a filtrovanie lyžiarskych stredísk na slovensku.

- Filtrovanie podľa typu lanovky.
- Nájdenie lyžiarskych stredísk vo vybranom okruhu (možnosť nastavenia okruhu).
- Nájdenie hotelov, alebo chát v okolí stredísk ktoré sa nachádzajú v oblasti (možnosť nastavenia max. vzdialenosti od strediska).
- Filtrovanie podľa výšky snehu v stredisku (dáta zo SHMU pomocou web scraping).
- Zobrazenie snehovej heatmapy.

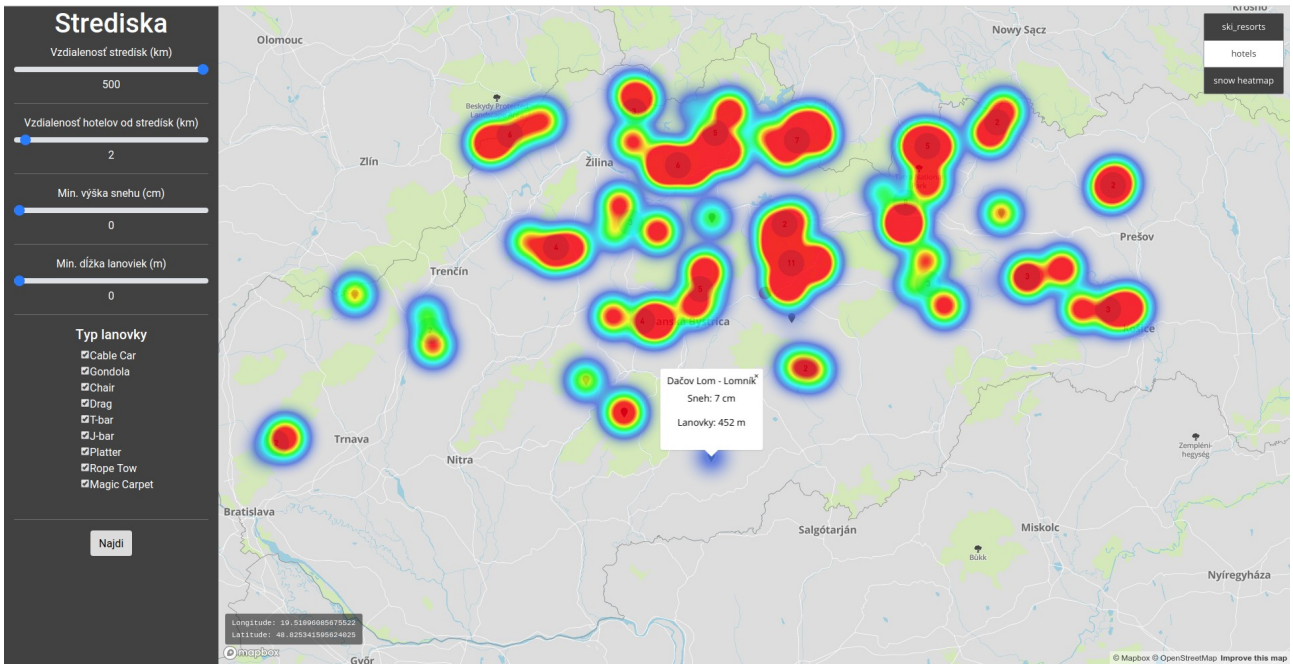
2. Dáta

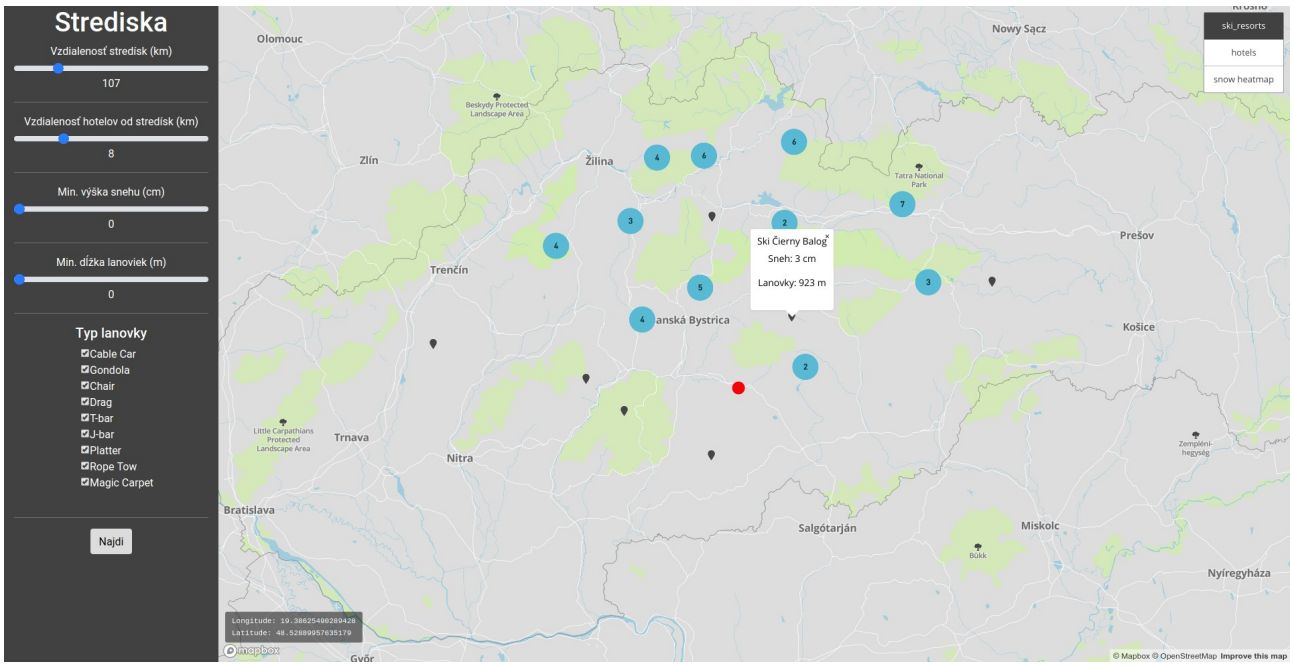
1. zdroj – openstreetmap – Slovensko
2. zdroj – SHMU web scraping (http://www.shmu.sk/sk/?page=68)

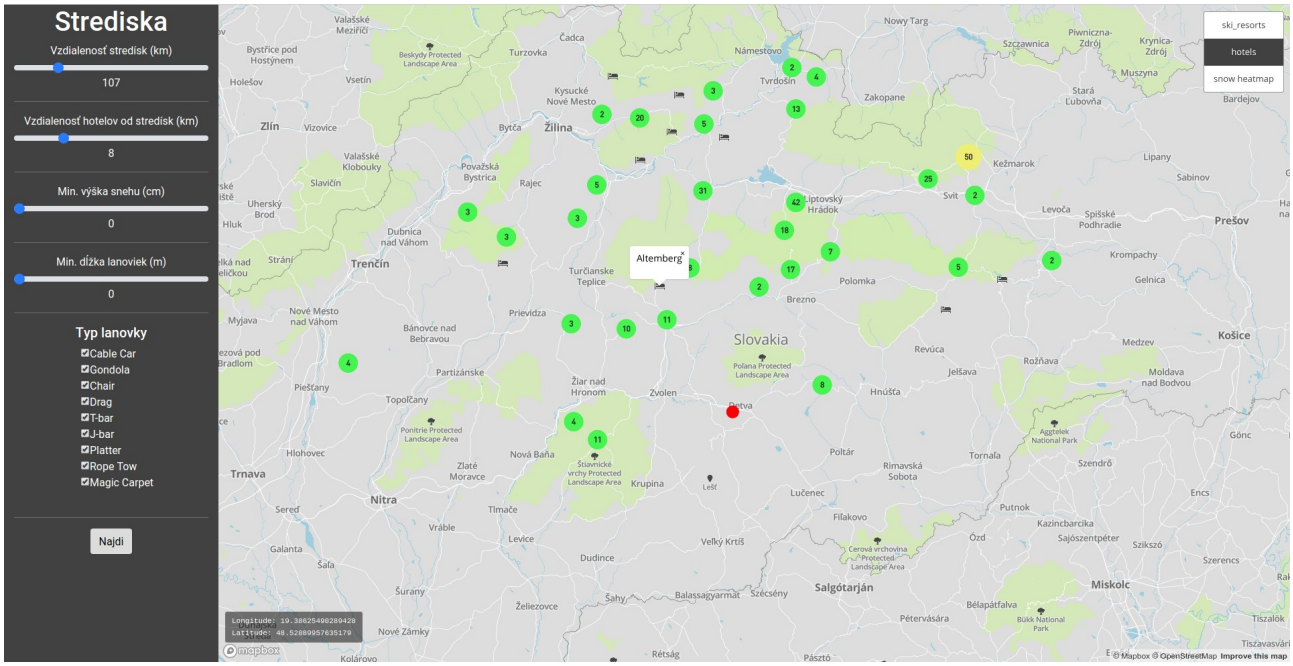
- Dáta zo SHMU sa uložia do csv.
- Pomocou webovej služby *locationiq.com* sa k lyžiarským strediskám nájdu súradnice.
- Dáta sa uložia do DB (meno strediska, výška snehu, súradnice).
- Na update snehových dát slúži script ktorý updatne DB.

3. Frontend

Na FE sa používa Mapbox GL. Poskytuje takú funkcionalitu aká je napísaná v prehľade. Ak je veľa hotelov alebo lyžiarskych stredísk nakope, tak sa zhlukujú. Na mape sa zobrazujú buď hotely, alebo lyžiarské strediská – kvôli prehladu. Hatmapa sa dá spustiť aj pri zobrazovaní hotelov a aj pri lyžiarskych strediskách. Po kliknutí na stredisko sa zobrazia podrobnejšie informácie – názov, sneh, dĺžka lanoviek. Po kliknutí na hotel sa zobrazí názov hotelu. Červený bod sa dá presúvať po mape a používa sa pri počítaní okruhu.
Na dynamický update dát sa používa ajax.







4. Backend

Na BE sa používa Python spolu s frameworkom Flask. Na pripojenie k DB sa používa *psycopg2*. Pri spustení aplikácie sa zavolá routa ktorá vráti na FE dáta o hoteloch a lyžiarskych strediskách v 50 km okolí defaultne nastaveného bodu.
Druhá routa sa používa keď používateľ klikne na tlačítko vyhľadať – na FE sa vrátia údaje podľa toho aké podmienky filtrovania si používateľ zvolil.

5. Databáza

V DB sú vytvorené nové tabuľky oproti defaultným z openstreetmap:

- **ski_aerialways**: všetky lanovky z openstreetmap + nový stĺpec v ktorom je dĺžka lanovky
- **shmu_ski**: lyžiarske strediská zo SHMU – súradnice, sneh, názov.
- **hotels**: hotely z openstreetmap + stĺpce pre súradnice lat a lon (kvôli tomu, že hotely zobrazujeme na mape ako bod, v osm sú ako plocha – nemusí sa to prepočítavať v každej query keď sa to vypočíta a uloží pri vytváraní tabuľky).
- **centers_with_aerialways**: Všetky lanovky v okolí 100 metrov od strediska sú priradené k tomuto stredisku (niektoré strediská sú blízko pri sebe, pri väčšej vzdialenosti už bolo priradene chybné). Obsahuje stĺpce - názov lyžiarskeho strediska, geometria strediska, geometria lanovky, tagy lanovky, sneh, súradnice, dĺžka lanovky.

Vytvorenie hotelov

```
CREATE TABLE hotels AS
SELECT name, tags, way FROM planet_osm_point WHERE tourism='hotel' or tourism = 'chalet'
UNION
SELECT name, tags, way FROM planet_osm_polygon WHERE tourism='hotel' or tourism='chalet'
```

Vytvorenie lanoviek

```
SELECT name, aerialway, landuse, leisure, sport, tags, way INTO ski_aerialways FROM planet_osm_line WHERE
aerialway = 'cable_car' or aerialway = 'gondola' or aerialway = 'chair_lift' or aerialway = 'mixed_lift' or aerialway =
'drag_lift' or aerialway = 't-bar' or aerialway = 'j-bar' or aerialway = 'platter' or aerialway = 'rope_tow' or aerialway
= 'magic_carpet'
```

Pridanie dĺžky lanoviek

```
UPDATE ski_aerialways SET aerialway_length = round(ST_LENGTH(way::geography))
```

Vytvorenie centier s lanovkami

```
CREATE TABLE centers_with_aerialways AS
SELECT shmu_ski.name as ski_center_name , shmu_ski.geom as ski_center_geom, aerialway.way as aerialway_geom,
aerialway.tags as aerialway_tags, shmu_ski.snow as snow,
shmu_ski.lat as lat, shmu_ski.lon as lon, aerialway.aerialway_length as aerialway_length
FROM ski_aerialways as aerialway INNER JOIN shmu_ski
ON ST_DWithin(aerialway.way, shmu_ski.geom, 0.005)
```

Pridanie súradníc do hotelov

```
UPDATE hotels SET lon = ST_X(ST_TRANSFORM(ST_Centroid(hotels.way),4674))
UPDATE hotels SET lat = ST_Y(ST_TRANSFORM(ST_Centroid(hotels.way),4674))
```

6. Query + Optimalizácia

Pôvodné query na strediská:

SELECT ski_center_name, snow, lon as long, lat, sum(aerialway_length) as aerialways_length FROM centers_with_aerialways WHERE
*ST_Distance_Sphere(ski_center_geom, ST_MakePoint(18.104103, 48.626156)) <= 100 * 1000*
AND snow >= 0 GROUP BY ski_center_name, lon, lat, snow HAVING sum(aerialway_length) >= 800

	QUERY PLAN text
1	GroupAggregate (cost=208.98..209.00 rows=1 width=56)
2	Group Key: ski_center_name, snow, ski_center_geom
3	Filter: (sum(aerialway_length) >= 400)
4	-> Sort (cost=208.98..208.98 rows=1 width=60)
5	Sort Key: ski_center_name, snow, ski_center_geom
6	-> Seq Scan on centers_with_aerialways (cost=0.00..208.97 rows=1 width=60)
7	Filter: ((snow >= 0) AND ((aerialway_tags -> 'aerialway'::text) = ANY ('{chair_lift,t-bar}

Optimalizácia query na strediská:

Vytvorenie indexu:

CREATE INDEX ski_center_geom_index ON centers_with_aerialways USING GIST (geography(ski_center_geom))

Upravenie ST_Distance_Sphere na ST_DWithin, pridanie ::geography:

SELECT ski_center_name, ski_center_geom FROM centers_with_aerialways WHERE
*ST_DWithin(ski_center_geom::geography, ST_SetSRID(ST_MakePoint(18.104103,48.626156), 4326)::geography, 100*1000)*
AND aerialway_tags->'aerialway'IN ('chair_lift', 't-bar') AND snow >=0 GROUP BY ski_center_name, snow, ski_center_geom HAVING sum(aerialway_length) >= 400

	QUERY PLAN text
1	GroupAggregate (cost=33.60..33.63 rows=1 width=56)
2	Group Key: ski_center_name, snow, ski_center_geom
3	Filter: (sum(aerialway_length) >= 400)
4	-> Sort (cost=33.60..33.61 rows=1 width=60)
5	Sort Key: ski_center_name, snow, ski_center_geom
6	-> Bitmap Heap Scan on centers_with_aerialways (cost=4.53..33.59 rows=1 width=60)
7	Recheck Cond: ((ski_center_geom)::geography && '0101000020E61000005D6A847EA61A3240DB183BE125504840'::geography)
8	Filter: ((snow >= 0) AND ((aerialway_tags -> 'aerialway'::text) = ANY ('{chair_lift,t-bar}'::text[])) AND ('0101000020E61000005D6A847EA61A
9	-> Bitmap Index Scan on ski_center_geom_index (cost=0.00..4.53 rows=52 width=0)
10	Index Cond: ((ski_center_geom)::geography && '0101000020E61000005D6A847EA61A3240DB183BE125504840'::geography)

Query na hotely:

WITH ski_centers_in_radius AS (
SELECT ski_center_name, ski_center_geom FROM centers_with_aerialways
*WHERE ST_DWithin(ski_center_geom::geography, ST_SetSRID(ST_MakePoint(18.104103,48.626156), 4326)::geography, 100*1000) AND aerialway_tags->'aerialway'IN ('chair_lift', 't-bar') AND snow >=0 GROUP BY ski_center_name, snow, ski_center_geom HAVING sum(aerialway_length) >= 400)*
SELECT DISTINCT ON(hotels.name) hotels.name, hotels.lon , hotels.lat FROM hotels
INNER JOIN ski_centers_in_radius ON ST_DWithin(ski_centers_in_radius.ski_center_geom, hotels.way, 20)

	QUERY PLAN text
1	Unique (cost=548.78..548.78 rows=1 width=21)
2	CTE ski_centers_in_radius
3	-> GroupAggregate (cost=33.60..33.63 rows=1 width=56)
4	Group Key: centers_with_aerialways.ski_center_name, centers_with_aerialways.snow, centers_with_aerialways.ski_center_geom
5	Filter: (sum(centers_with_aerialways.aerialway_length) >= 400)
6	-> Sort (cost=33.60..33.61 rows=1 width=60)
7	Sort Key: centers_with_aerialways.ski_center_name, centers_with_aerialways.snow, centers_with_aerialways.ski_center_geom
8	-> Bitmap Heap Scan on centers_with_aerialways (cost=4.53..33.59 rows=1 width=60)
9	Recheck Cond: ((ski_center_geom)::geography && '0101000020E61000005D6A847EA61A3240DB183BE125504840'::geography)
10	Filter: ((snow >= 0) AND ((aerialway_tags -> 'aerialway'::text) = ANY ('{chair_lift,t-bar}'::text[])) AND ('0101000020E61000005D6A847EA61A3240DB1
11	-> Bitmap Index Scan on ski_center_geom_index (cost=0.00..4.53 rows=52 width=0)
12	Index Cond: ((ski_center_geom)::geography && '0101000020E61000005D6A847EA61A3240DB183BE125504840'::geography)
13	-> Sort (cost=515.15..515.16 rows=1 width=21)
14	Sort Key: hotels.name
15	-> Nested Loop (cost=0.00..515.14 rows=1 width=21)
16	Join Filter: ((ski_centers_in_radius.ski_center_geom && st_expand(hotels.way, '20'::double precision)) AND (hotels.way && st_expand(ski_centers_in_ra
17	-> CTE Scan on ski_centers_in_radius (cost=0.00..0.02 rows=1 width=32)
18	-> Seq Scan on hotels (cost=0.00..136.04 rows=1404 width=157)

Optimalizácia query na hotely:

Vytvorenie indexu:

CREATE INDEX hotel_way_index ON hotels USING GIST (geometry(way))

	QUERY PLAN text
1	Unique (cost=42.09..42.09 rows=1 width=21)
2	CTE ski_centers_in_radius
3	-> GroupAggregate (cost=33.60..33.63 rows=1 width=56)
4	Group Key: centers_with_aerialways.ski_center_name, centers_with_aerialways.snow, centers_with_aerialway...
5	Filter: (sum(centers_with_aerialways.aerialway_length) >= 400)
6	-> Sort (cost=33.60..33.61 rows=1 width=60)
7	Sort Key: centers_with_aerialways.ski_center_name, centers_with_aerialways.snow, centers_with_aerialwa...
8	-> Bitmap Heap Scan on centers_with_aerialways (cost=4.53..33.59 rows=1 width=60)
9	Recheck Cond: ((ski_center_geom)::geography && '0101000020E61000005D6A847EA61A3240DB183BE...
10	Filter: ((snow >= 0) AND ((aerialway_tags -> 'aerialway'::text) = ANY ('{chair_lift,t-bar}'::text[])) AND ('010...
11	-> Bitmap Index Scan on ski_center_geom_index (cost=0.00..4.53 rows=52 width=0)
12	Index Cond: ((ski_center_geom)::geography && '0101000020E61000005D6A847EA61A3240DB183B...
13	-> Sort (cost=8.46..8.47 rows=1 width=21)
14	Sort Key: hotels.name
15	-> Nested Loop (cost=0.15..8.45 rows=1 width=21)
16	-> CTE Scan on ski_centers_in_radius (cost=0.00..0.02 rows=1 width=32)
17	-> Index Scan using hotel_way_index on hotels (cost=0.15..8.42 rows=1 width=157)
18	Index Cond: (way && st_expand(ski_centers_in_radius.ski_center_geom, '20'::double precision))
19	Filter: ((ski_centers_in_radius.ski_center_geom && st_expand(way, '20'::double precision)) AND _st_dwit...