Modul Data statistics

Tím PARKety, č. 15

Vedúci projektu: Ing. Ivan Srba, PhD.

Predmet: Tímový projekt I

Ročník: 2017/2018

Vypracoval(i): Samuel Púčať, Martin Hoang, Jakub Mičo, Jakub Hučko, Miroslav Lehotský, Stanislav Vnenčák

Babinec Peter, Bc. Hoang Martin, Bc.
Hučko Jakub, Bc. Karas Marek, Bc.
Lehotský Miroslav, Bc. Mičo Jakub, Bc.
Púčať Samuel, Bc. Vnenčák Stanislav, Bc.

11. mája 2018

Obsah

Zoznam obrázkov

1 Analýza

Modul Data statistics zodpovedá za počítanie, ukladanie a zobrazovanie štatistík o parkoviskách a parkovacích miestach:

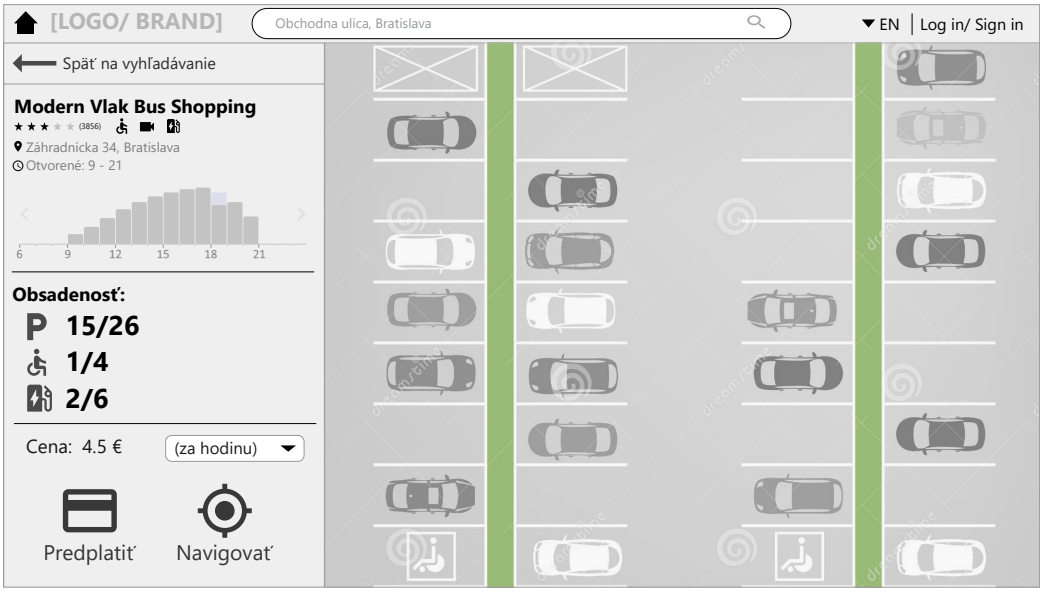
- *štatistiky obsadenosti parkovísk,*
- *štatistiky obsadenosti parkovacích miest,*
- *štatistiky vyťaženosti parkovísk,*
- *štatistiky vyťaženosti parkovacích miest,*
- *štatistiky celkového a priemerného počtu áut na parkoviskách,*
- *štatistiky celkového a priemerného počtu áut na parkovacích miestach,*
- *štatistiky celkovej a priemernej doby státia na parkovisku,*
- *štatistiky celkovej a priemernej doby státia na parkovacích miestach,*
- *štatistiky prekročeného státia na parkoviskách,*
- *štatistiky prekročeného státia na parkovacích miestach.*

2 Návrh

2.1 UX návrh zobrazenia štatistík

2.1.1 Zobrazenie štatistiky obsadenosti parkoviska

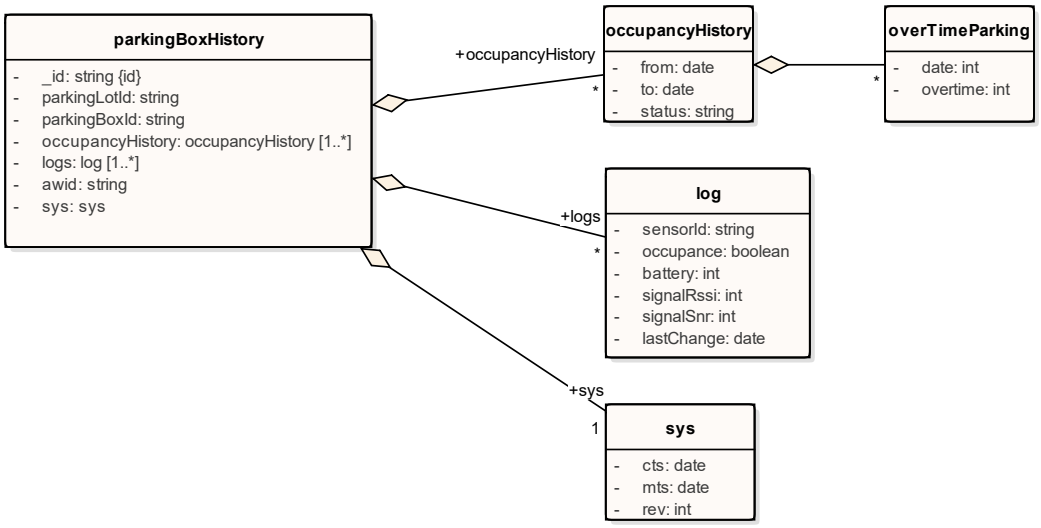
Na obrázku 2.1 vidieť návrh zobrazenia štatistiky obsadenosti parkoviska v ľavom paneli. Štatistika obsadenosti je zobrazená ako graf závislosti času a percentuálnej obsadenosti pre každý hodinový interval aktuálneho dňa.



Obr. 2.1: *Zobrazenie štatistiky obsadenosti parkoviska*

2.2 Model údajov pre entitu histórie parkoviska

Štatistiky je potrebné počítať z historických údajov. Ukladaná je každá zmena obsadenosti senzora. Na obrázku 2.2 je znázornený návrh modelu údajov pre ukladanie histórie obsadenosti parkovacích miest.



Obr. 2.2: *Model údajov pre ukladanie histórie obsadenosti parkovacích miest*

Každý záznam histórie parkovacieho miesta má:

- *parkingLotId* - referencia na parkovisko
- *parkingBoxId* - referencia na parkovacie miesto
- *occupancyHistory* - pole histórie obsadenosti s údajmi pre každý element:
  - *from* - dátum a čas začiatku stavu obsadenosti
  - *to* - dátum a čas konca stavu obsadenosti
  - *status* - obsadenosť (voľné, obsadené)
  - *overtimeParking* - pole časov nad povolený limit
    - *date* - dátum státia nad povolený limit
    - *overtime* - čas státia nad povolený limit v milisekundách
- *logs* - pole všetkých prijatých správ zo senzorov parkovacieho miesta s údajmi pre každý element:
  - *sensorId* - referencia na senzor
  - *occupance* - obsadenosť

– *battery* - stav batérie

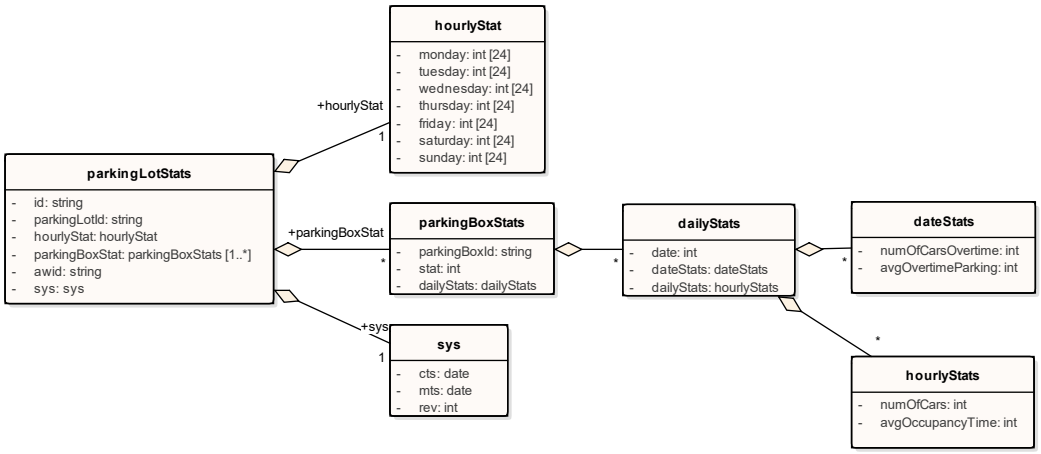
– *signalRssi* - sila signálu senzora (receiver signal strength)

– *signalSnr* - pomer signálu k šumu (signal to noise ratio)

– *lastChange* - dátum a čas poslednej zmeny

2.3 Model údajov pre entitu štatistík parkoviska

Štatistiky parkoviska sa po pravidelnom výpočte ukladajú do databázy. Na obrázku 2.3 je znázornený návrh modelu údajov pre ukladanie štatistík parkoviska.



Obr. 2.3: *Model údajov pre ukladanie štatistík parkoviska*

Každý záznam štatistík parkoviska má:

- *parkingLotId* - referencia na parkovisko
- *hourlyStat* - zoznam percentuálnych hodnôt obsadenosti pre každú hodinu každého dňa v týždni
- *parkingBoxStat* - štatistiky pre parkovacie miesta
  - *parkingBoxId* - id parkovacieho miesta
  - *stat*

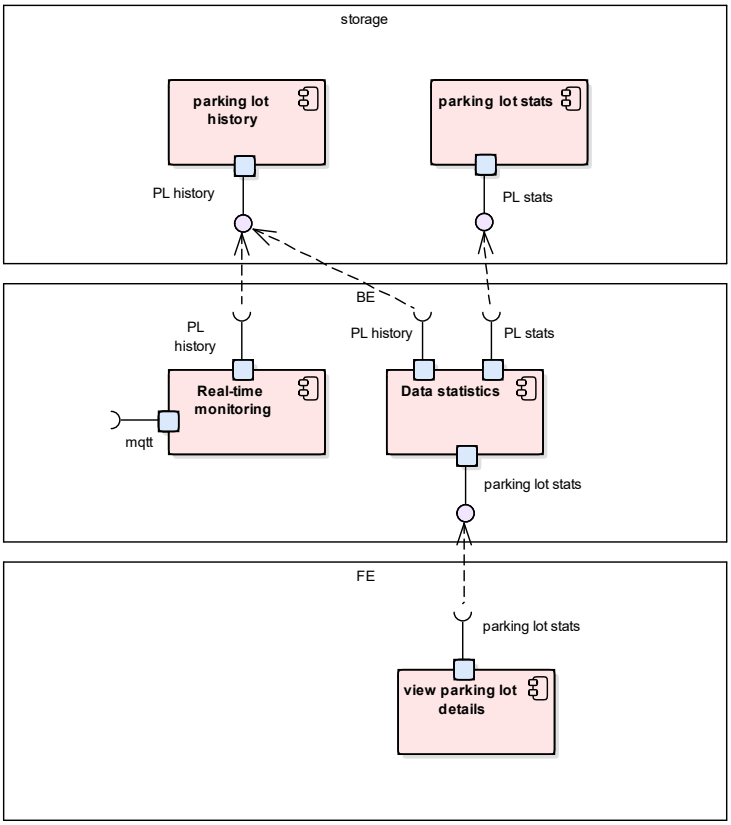
  - *dailyStats* - pole posledných 90 dní
    - *date*
    - *dateStats*
      - *numOfCarsOvertime* - počet vozidiel stojacich nad povolené minúty
      - *avgOvertimeParking* - priemerný čas státia vozidiel stojacich nad povolené minúty
    - *hourlyStats* - pole veľkosti 24
      - *numOfCars* - počet vozidiel na parkovisku v danej hodine
      - *avgOccupancyTime* - priemerný čas státia

- *awid*
- *sys*

3 Implementácia

3.1 Výrez z high level architektúry

Na obrázku 3.1 sú znázornené komponenty high level architektúry spolupracujúce s týmto modulom.

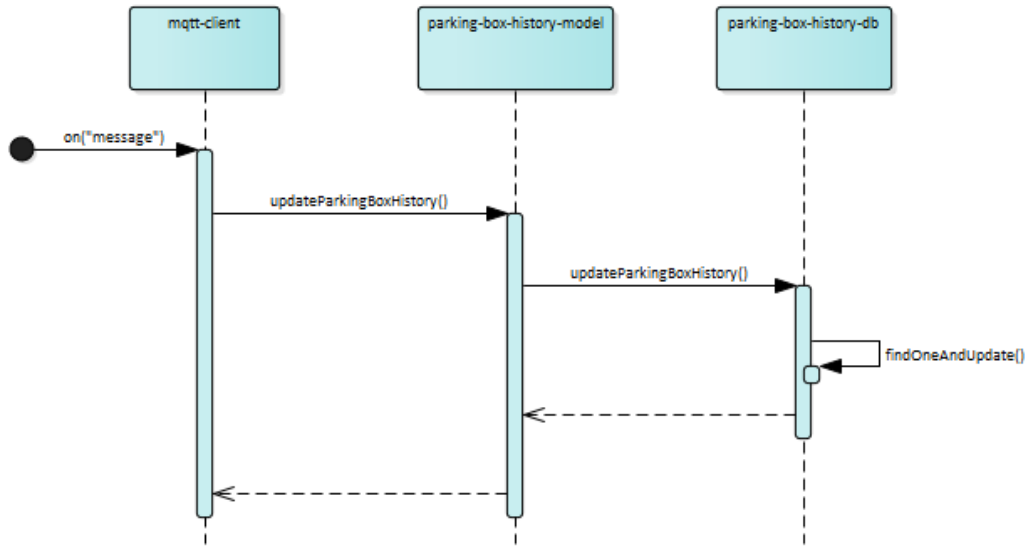


Obr. 3.1: *Celkový náhľad na architektúru*

3.2 Ukladanie histórie parkovísk

Pri komunikácii s parkovacími senzormi je potrebné ukladať do databázy históriu parkovacích miest. Na obrázku 3.2 je znázornené ukladanie histórie

parkovacích miest do databázy.



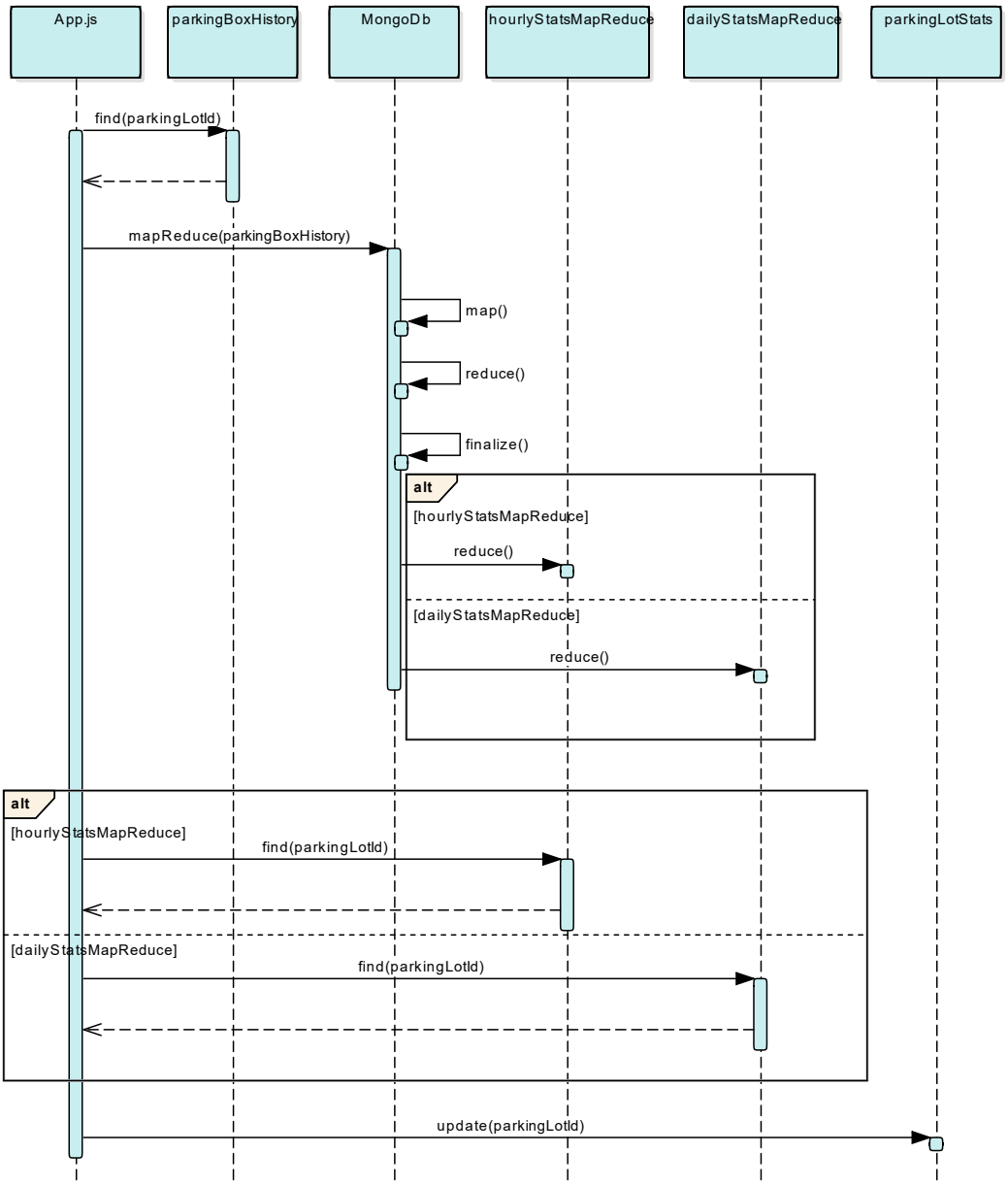
Obr. 3.2: *Sekvenčný diagram ukladania histórie parkovacích miest*

Kolekcia *parkingBoxHistory* v databáze obsahuje pre každé z parkovacích miest unikátny záznam, ktorý obsahuje jednoznačné identifikátory *parkingLotId* a *parkingBoxId* a polia pre ukladanie histórie (pozri časť 2.2).

Pri prijatí každej novej správy sa uložia extrahované dáta z tejto správy do poľa *logs* pre relevantné parkovacie miesto v kolekcii *parkingBoxHistory*. Modul *mqtt* sa pred zavolaním metódy *updateParkingBoxHistory* postará o kontrolu, či sa zmenil aj stav obsadenosti parkovacieho miesta. Pokiaľ áno, spolu s logom sa doplnia do histórie tiež informácie o zmene obsadenosti parkovacieho miesta a to do poľa *occupancyHistory*.

3.3 Počítanie štatistík parkoviska

Štatistiky sú počítané v pravidelných časových intervaloch (raz za deň) pomocou inkrementálneho map-reduce pre všetky parkoviská a vypočítané štatistiky sú uložené do databázy. Na obrázku 3.3 je znázornený proces počítania a ukladania štatistík.



Obr. 3.3: *Sekvenčný diagram počítania štatistík parkoviska*

3.3.1 Výpočet hourlyStatsMapReduce

Ide o kolekciu v databáze, z ktorej sa počítajú hodinové štatistiky priemernej obsadenosti počas týždňa (viď. obrázok 3.10). Táto kolekcia je aktualizovaná každý deň pomocou algoritmu map-reduce. Algoritmus map-reduce pozostáva

z troch krokov, *map*, *reduce* a *finalize*.

Mapovacia funkcia vytvára páry kľúč-hodnota, v tomto prípade kľúč pozostáva z *parkingBoxId*, *parkingLotId*, *hour* (index hodiny 0 - 23) a *dayNm* (index dňa 0 - 6). Hodnota je tvorená z počtu minút za danú hodinu. Ako môžeme vidieť na obrázku 3.4, map-reduce algoritmus prebieha v databáze na servery, pre každý *occupancyHistory* skontroluje či patrí do intervalu a potom logy rozdelí na dni. Potom pre každý deň rozdelí na hodiny a tie hodiny sa pomocou emitovaciej funkcie odosielajú v tvare kľúč-hodnota.

Reduce funkcia pre všetky vytvorené polia rovnakých hodnôt kľúča, zráta všetky prvky v poli.

Nakoniec finalize funkcia vracia 1 k ide o hodnotu väčšiu ako 390 (táto hodnota je polovica možného času za obdobie 3 mesiace = ~13 týždňov ~780 hodín polovica z toho je 390 hodín), ináč vracia 0. Tieto údaje sa aktualizujú do kolekcie hourlyStatsMapReduce. Nakoniec sa z tejto kolekcie vytvorí histogram, ktorý ukladá do kolekcie parkingLotStats.

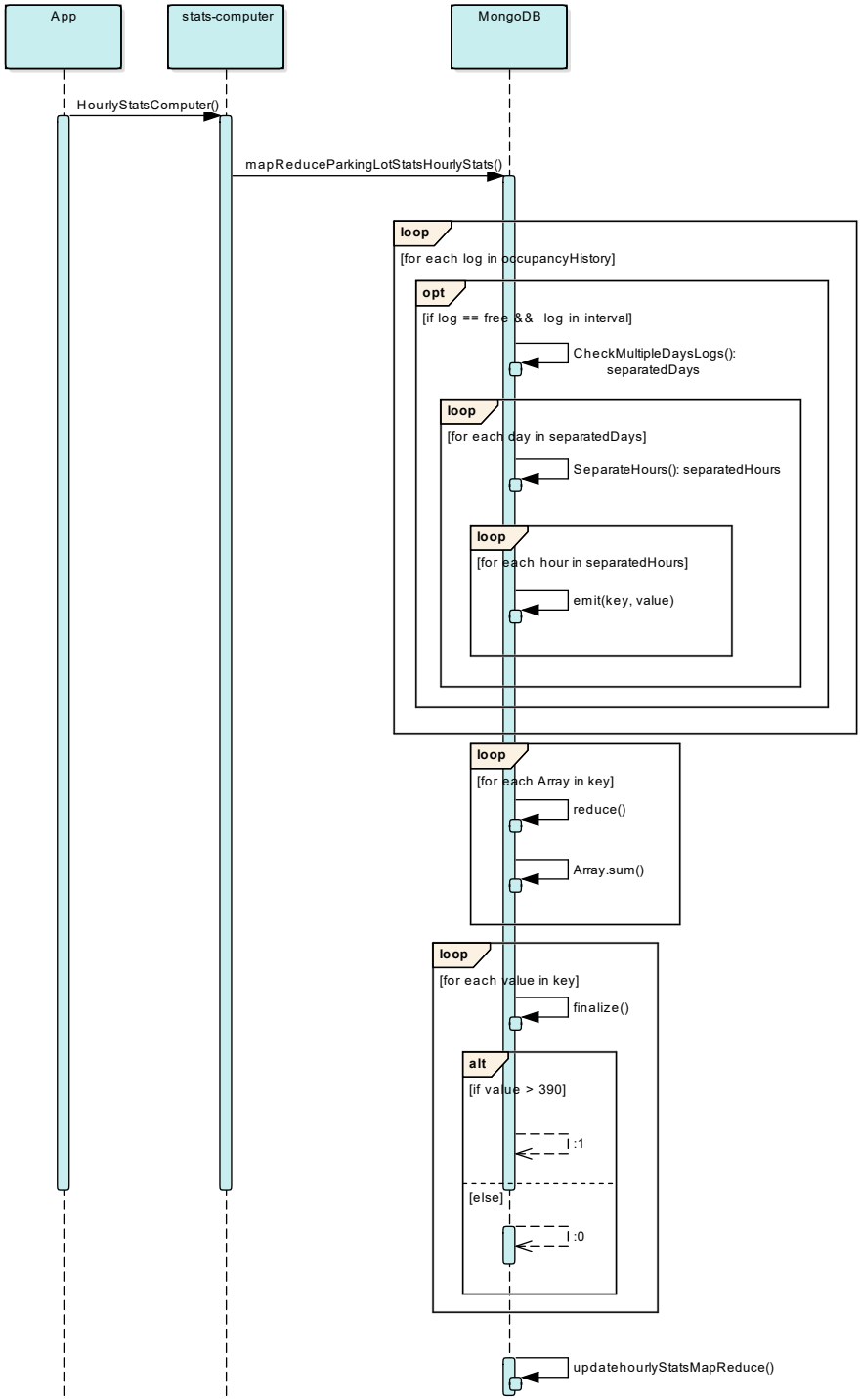
3.3.2 Výpočet dailyStatsMapReduce

Taktiež ako hourlyStatsMapReduce sa kolekcia dailyStatsMapReduce počíta pomocou map-reduce algoritmu. Princíp je podobný, preto ho ani nebudeme kresliť v UML notácií.

Mapovacia funckia sa vytvára páry kľúč-hodnota, v tomto prípade kľúč pozostáva z *parkingBoxId*, *parkingLotId*, *date* a *hour* (index hodiny 0 - 23). Hodnota je tvorená z počtu minút za danú hodinu.

Reduce funkcia spočíta všetky prvky poli a ich sumu, z čoho vytvorí novú hodnotu.

Nakoniec finalize funkcia skontroluje, či hodnota daného kľúča je objekt, nakoľko reduce funkciou idú iba kľúče, ktoré majú čo redukovať. Ak nejde o objekt tak ho vytvorí. Nakoniec prepočíta hodnoty na milisekundy a aktualizuje databázu dailyStatsMapReduce. Z tejto kolekcie sa potom počítajú hourlyStats v parkingLotStats kolekcií ale na úrovní každého parkingBoxu.

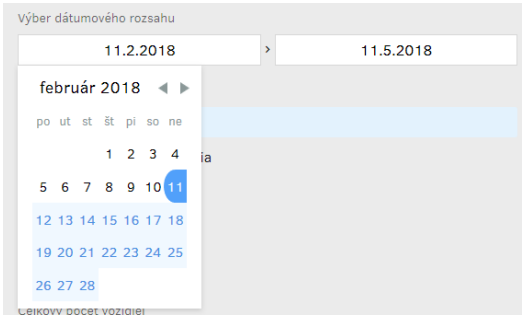


Obr. 3.4: *Sekvenčný diagram počítania hourlyStatsMapReduce pomocou map-reduce algoritmu.*

3.4 Načítanie štatistík

3.4.1 Výber dátumového rozsahu

V historickom zobrazení parkoviska je možné meniť rozsah dátumov, pre ktorý sa zobrazia historické štatistiky. Na implementáciu boli použité komponenty *<DayPickerInput/>*, pri načítaní stránky je horná hranica nastavená na aktuálny deň a dolná hranica na deň pred 3 mesiacmi (viď. obr. 3.5).



Obr. 3.5: *Výber dátumového rozsahu*

3.4.2 Agregované štatistiky pre parkovisko

API

- */getAvgNumOfCarsForDayAllBoxes* - pre všetky parkovacie boxy v rámci parkoviska, vráti priemerné štatistiky pre počet vozidiel a čas obsadenosti pre daný deň a pre dané hodiny za posledné 3 mesiace. Napríklad pre všetky utorky, v časovom intervale od 12:00 - 16:00 za posledné 3 mesiace.

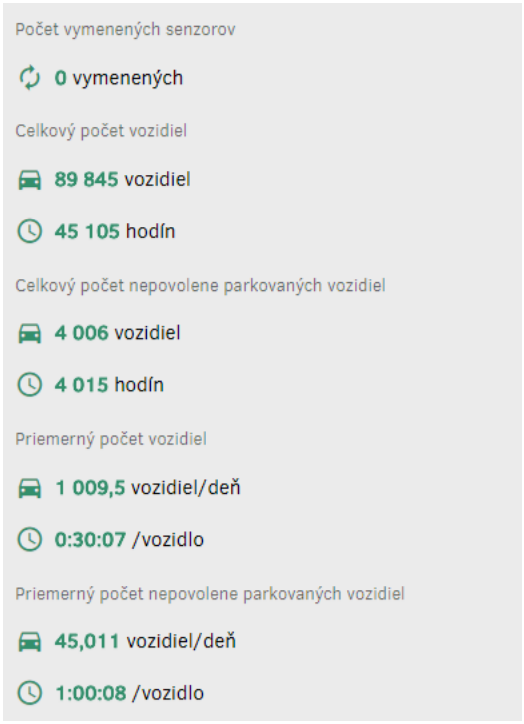
- */getAvgNumOfCarsForDay* - pre parkovací box vráti priemerný počet vozidiel a priemernú dobu obsadenosti, pre všetky hodiny v rozsahu a pre daný deň, napríklad utorky, v intervale od-do za posledné 3 mesiace. Jedná sa o rovnakú api ako /getAvgNumOfCarsForDayAllBoxes, akurát pre konkrétny parkingBox.

- */getTotalAndAvgOccupanceAllBoxes* - pre parkovisko a všetky parko-

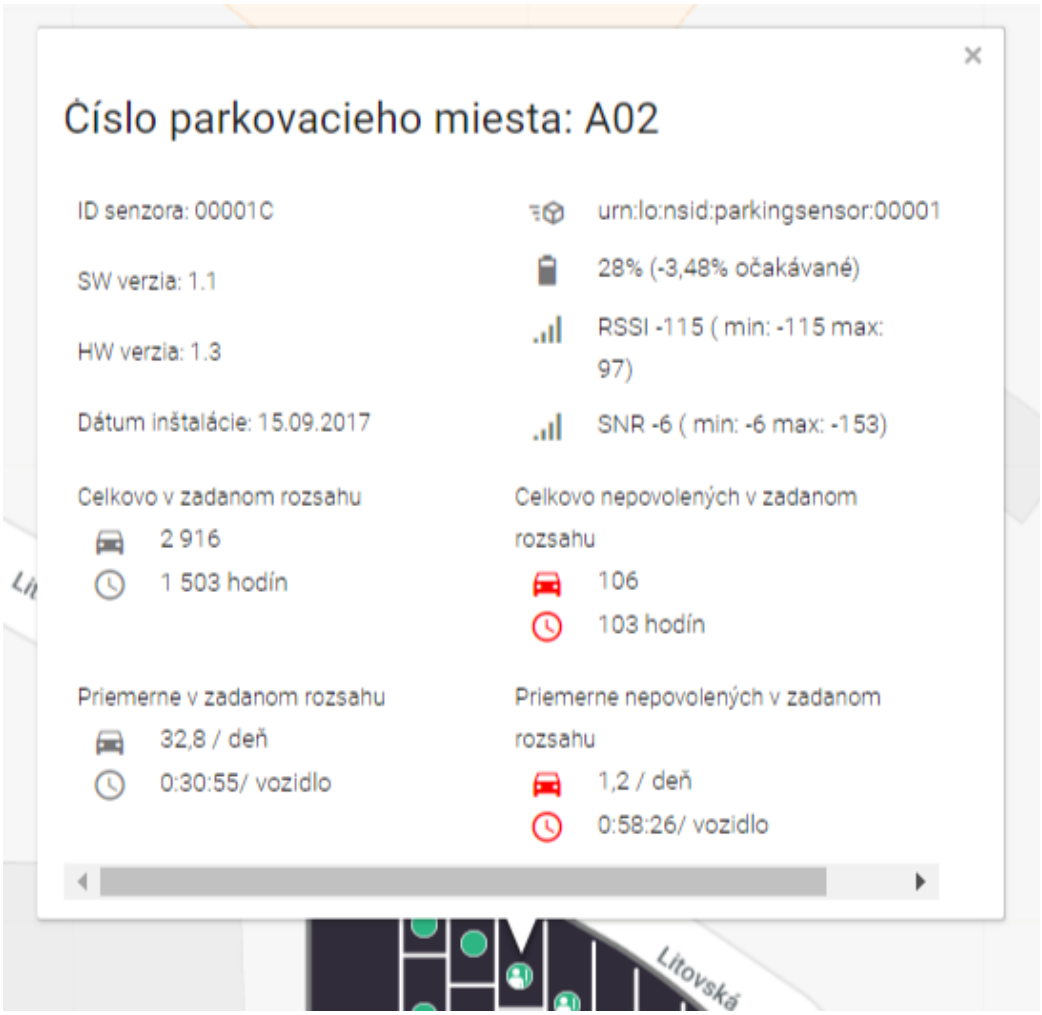
vacie miesta v rámci parkoviska, vráti komplexné štatistiky, menovite: celkový počet vozidiel, celkový čas obsadenosti, priemerný čas obsadenosti, priemerná percentuálna obsadenosť, celkový počet vozidiel nad voľné minúty, celkový čas obsadenosti nad voľné minúty, priemerný čas obsadenosti nad voľné minúty, minimálne a maximálne hodnoty signálov za daný časový interval pre každý parkingBox a tiež pokles batérie za daný časový interval pre každý parkingBox.

Zobrazenie

Vrátené hodnoty sa následne zobrazujú v ľavom paneli (viď. obr. 3.6) pre celé parkovisko, alebo v info okne nad senzorom pre parkovacie miesto a senzor zároveň (viď. obr. 3.7).



Obr. 3.6: *Štatistiky celého parkoviska zobrazené na ľavom paneli*



Obr. 3.7: *Štatistiky parkovacieho miesta zobrazené pomocou infowindow*

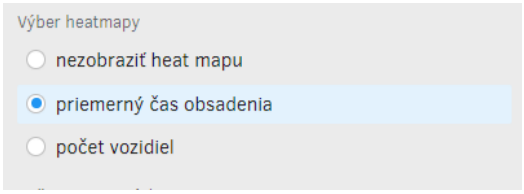
3.4.3 Heat mapa

API

- */getAvgNumOfCarsPerIntervalAllBoxes* - pre všetky parkovacie boxy, v rámci parkoviska za daný interval, vráti priemerné štatistiky pre počet vozidiel a čas obsadenosti (hodnoty pre graf).

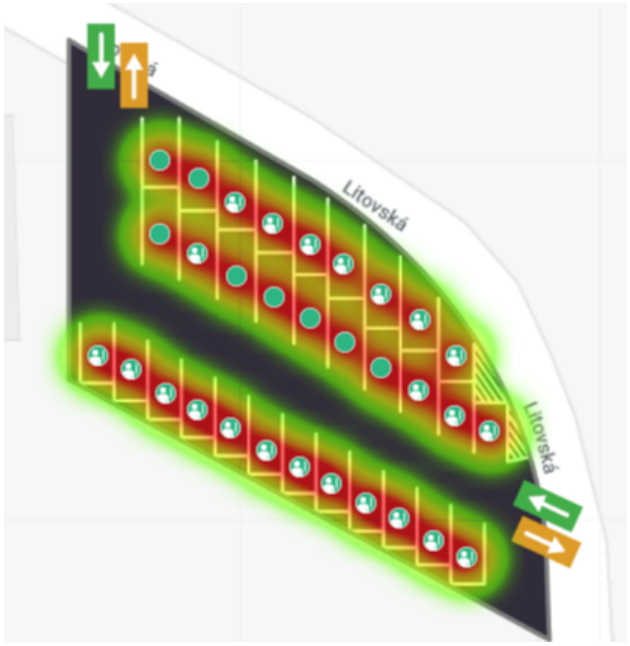
Zobrazenie

Výber heatmapy je realizovaný pomocou komponentu *UU5.Forms.Radios*. Je možné vybrať z troch možností, ktoré je možné vidieť na obrázku 3.8.



Obr. 3.8: *Výber heat mapy*

Následne sa vizualizácia realizuje pomocou komponentu *HeatmapLayer* knižnize *react-google-maps*. Tento komponent obaľuje funkcionalitu vytvárania heat máp, ktoré Google Mapy priamo obsahujú.



Obr. 3.9: *Heat mapa*

3.4.4 Štatistiky zobrazované v grafoch

API

API endpoint-y pre celé parkovisko:

- */getHourlyAvgOccupancePerDayForParkingLot* - pre parkovisko, vráti priemernú dobu obsadenosti, pre každý deň v týždni a pre každú hodinu v danom dni.
- */getTotalNumOfCarsForDayAllBoxes* - pre každý parkovací box v rámci parkoviska, vráti celkový počet vozidiel pre každý deň v intervale od-do (hodnoty pre graf)
- */getParkingLotHistoryAvgVehicleCountPerDay* - pre parkovisko, vráti priemerný počet vozidiel pre každý deň v intervale od-do (hodnoty pre graf)
- */getParkingLotHistoryTotalParkingTimePerDay* - pre parkovisko, vráti celkovú dobu obsadenosti pre každý deň v intervale od-do.
- */getParkingLotHistoryAvgParkingTimePerDay* - pre parkovisko, vráti priemernú dobu obsadenosti pre každý deň v intervale od-do.
- */getParkingLotCarsNumOverduePerDay* - pre parkovisko, vráti celkový počet vozidiel, ktoré prekročili voľné minúty, pre každý deň v intervale od-do.
- */getParkingLotTotalOverdueTimePerDay* - pre parkovisko, vráti celkovú dobu obsadenosti nad voľné minúty, pre každý deň v intervale od-do.
- */getParkingLotAvgOverdueTimePerDay* - pre parkovisko, vráti priemernú dobu obsadenosti nad voľné minúty, pre každý deň v intervale od-do.

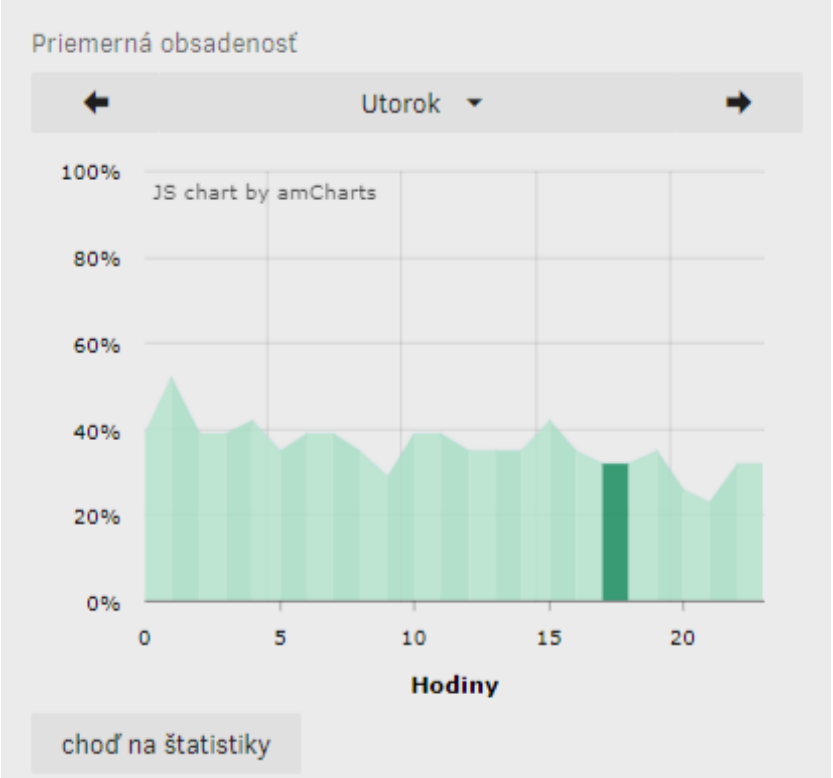
API endpoint-y pre jednotlivé boxy:

- */getTotalNumOfCarsForDay* - pre konkrétny parkovací box, vráti celkový počet vozidiel pre každý deň v intervale od-do (hodnoty pre graf);

- */getAvgNumOfCarsPerInterval* - pre parkovací box vráti priemerný počet vozidiel a priemernú dobu obsadenosti, pre všetky hodiny v časovom intervale od-do, max. však za posledné 3 mesiace;
- */getParkingBoxHistoryTotalParkingTimePerDay* - pre parkovací box, vráti celkovú dobu obsadenosti pre každý deň v intervale od-do (hodnoty pre graf);
- */getParkingBoxHistoryAvgParkingTimePerDay* - pre parkovací box, vráti priemernú dobu obsadenosti pre každý deň v intervale od-do (hodnoty pre graf);
- */getParkingBoxCarsNumOverduePerDay* -pre parkovací box, vráti celkový počet vozidiel, ktoré prekročili voľné minúty, pre každý deň v intervale od-do (hodnoty pre graf);
- */getParkingBoxTotalOverdueTimePerDay* - pre parkovací box, vráti celkovú dobu obsadenosti nad voľné minúty, pre každý deň v intervale od-do (hodnoty pre graf);
- */getParkingBoxAvgOverdueTimePerDay* - pre parkovací box, vráti priemernú dobu obsadenosti nad voľné minúty, pre každý deň v intervale od-do (hodnoty pre graf).

Zobrazenie

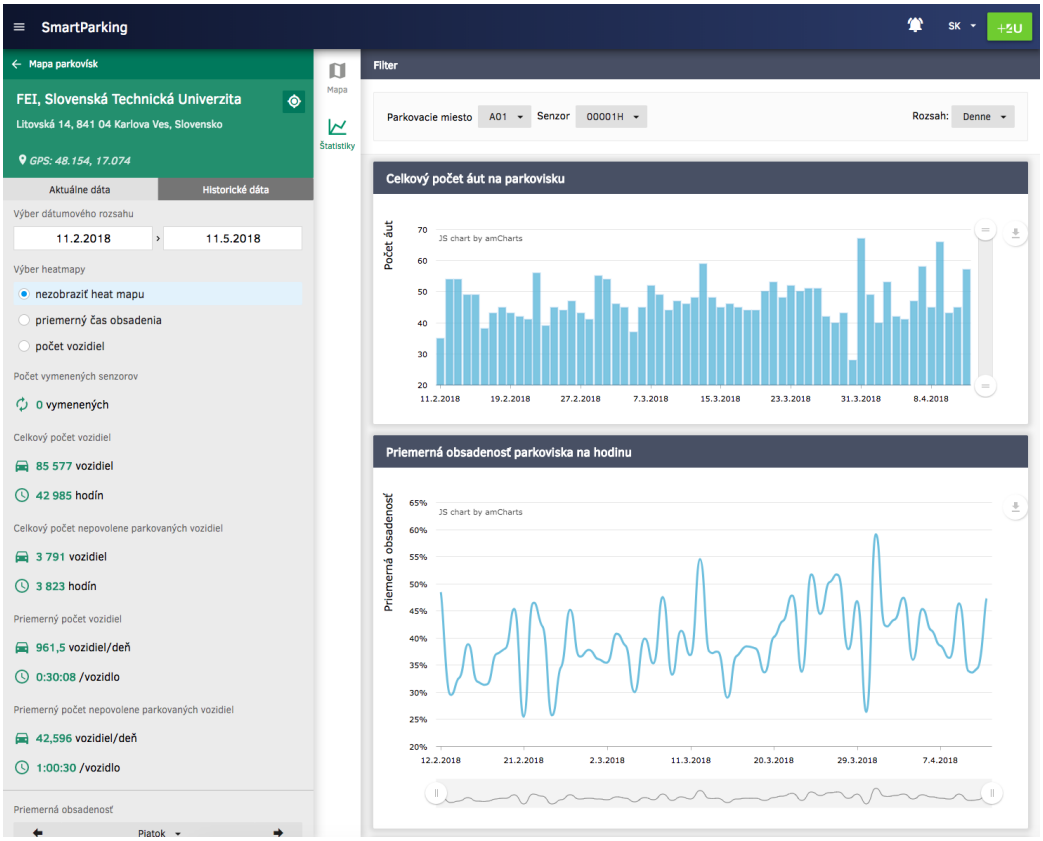
Pre zobrazenie grafov štatistík sa používa knižnica *am-charts*. Grafy sa zobrazujú buď v ľavom paneli (viď. obr. 3.10), alebo v samostatnom tab-e štatistík prístupnom v historickom zobrazení (viď. obr. 3.11)



Obr. 3.10: *Graf priemernej obsadenosti v ľavom paneli*

V rámci zobrazenia priemernej obsadenosti je možné zobraziť štatistické údaje pre každý z dňa v týždni výberom konkrétneho dňa zo zoznamu dní nad grafom alebo prepínaním dní pomocou navigačných tlačidiel.

Na základe aktuálnej hodiny je používateľovi výrazne vyznačená hodina, aby sa v grafe vedel jednoducho orientovať. Konkrétne informácie pre každú hodinu dňa sa používateľovi zobrazujú pri prechode nad samotným grafom.



Obr. 3.11: *Tab s grafmi*

Zobrazenie grafových štatistík sa riadi viacerými filtrami. Tieto štatistiky je možné filtrovať na základe:

- dátumu *OD* - zobrazí štatistiky pre zadaný dátum *OD*;
- dátumu *DO* - zobrazí štatistiky pre zadaný dátum *DO*;
- vybraného parkovacieho miesta - zobrazí štatistiky pre vybrané parkovacie miesto (na základe názvu parkovacieho miesta);
- vybraného senzora parkovacieho miesta - zobrazí štatistiky pre vybraný senzor parkovacieho miesta (v prípade viacerých senzorov na parkovacom mieste - momentálne neexistujú takéto *API*, ktoré by podporovali zobrazovanie parkovacích senzorov). Ak nie je vybraný žiaden, zobrazujú sa súhrnné štatistiky pre celé parkovisko;

- zvoleného rozsahu - zobrazí agregované štatistiky podľa zvoleného rozsahu (podporované rozsahy: denne, týždenne, mesačne).

Nakoľko knižnica použitá na zobrazovanie grafov podporuje interakciu s vygenerovanými grafmi, používateľ si môže grafy uložiť vo viacerých formátoch, môže do nich kresliť alebo ich môže ľubovoľne škálovať.

Pri pohybe nad jednotlivými grafmi sa používateľovi zobrazujú dodatočné informácie k jednotlivým dátam v grafe (dátum a hodnota) pre lepšiu čitateľnosť.

4 Testovanie

Testovanie softvéru pomáha predísť chybám, čím skorej sa chyba nájde, tým je menej nákladná jej oprava.

4.1 Testovanie map-reduce algoritmov

Testovanie map-reduce algoritmov sa testovalo viacerými spôsobmi a to priamo v node.js, kde sa dá imitovať mapovacia funkcia alebo pomocou prostredia *Studio 3T*[1].

4.1.1 Testovanie v node.js

Testovanie priamo pomocou node.js nám uľahčilo aj prostredie *Webstorm*[2], ktoré nám dovoľuje odladovať program po krokoch a taktiež sa pozerať na premenné v hociktorom kroku v programe. Testovanie v node.js prebiehalo prvotným napísaním algoritmov ako sú napríklad rozdelenie na roky, mesiace, dni, hodiny a potom následným vytvorením dát, ktoré by mohli predstavovať extrém pre tieto algoritmy.

Na ukážke 4.1 môžeme vidieť príklad na vytvorené dáta. Prvý príklad predstavuje extrém kedy sa log z parkingBoxHistory prebieha cez rok, mesiac aj dni. Ako sme už spomínali vďaka webstorm prostrediu sme mohli krokovať náš algoritmus a tým sa pozrieť na každý krok v programe, či sa vykonáva správne a vracia očakávanú hodnotu. Druhý príklad je na otestovanie viacerých logov počas jednej hodiny, testuje sa tým či sa logy správne rozložia a zaradia do patričnej skupiny.

[1] https://studio3t.com/

[2] https://www.jetbrains.com/webstorm/

```
occupancyHistory: [
  {
    from: new Date ('2017-11-29T16:03:32.000Z'),
    to: new Date ('2018-01-5T08:03:32.000Z'),
    status: "occupied",
    overtimeParking: {...}
  },
  {
    from: new Date ('2018-01-5T08:13:43.000Z'),
    to: new Date ('2018-01-5T08:23:52.000Z'),
    status: "occupied",
    overtimeParking: {...}
  },
  {
    from: new Date ('2018-01-5T08:54:43.000Z'),
    to: new Date ('2018-01-5T08:57:52.000Z'),
    status: "occupied",
    overtimeParking: {...}
  }
  ...
];
```

Ukážka 4.1: *Ukážka vytvorených dát pre testovacie účely*

4.1.2 Testovanie v prostredí Studio 3T

Studio 3T má viacero funkcionalít, jedná z nich je odladovanie krokov map-reduce algoritmu (map, reduce, finalize) na úrovni MongoDB. Táto možnosť sa využívala až potom ako sme otestovali na svojich vytvorených dátach, keď sme chceli testovať na reálnych dátach z databázy. Dovoľuje nám v každom kroku map-reduce algoritmu si pozrieť výsledok daného kroku a volať jednoduché dopyty nad týmito výsledkami.