Modul Real-time Monitoring

Tím PARKety, č. 15

Vedúci projektu: Ing. Ivan Srba, PhD.

Predmet: Tímový projekt I

Ročník: 2017/2018

Vypracoval(i): Martin Hoang, Samuel Púčať, Jakub Hučko,
Marek Karas

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 real-time monitoring sa zaoberá zobrazením informácií v reálnom čase. Umožňuje vyhľadať parkoviská podľa názvu alebo adresy a zobraziť parkoviská na mape s aktuálnym stavom na parkoviskách. Modul komunikuje so senzormi pomocou MQTT a Live Object API. Túto komunikáciu nám sprostredkováva Orange Slovensko. Ostatné údaje ako sú parking lots (parkoviska), PL img (obrázok parkoviska) a parking lot history (históriu parkoviska) získava z dátového úložiska, presnejšie z MongoDB.

2 Návrh

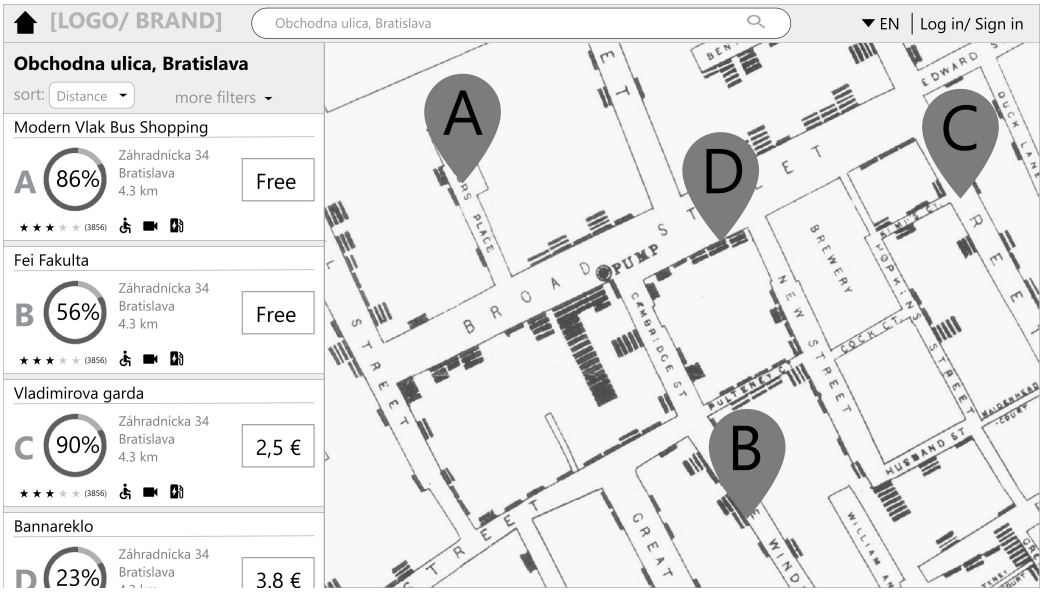
Implementácií predchádza návrh. Navrhovali sme viacero dátových entít ale aj UX obrazovky pre ľahšiu komunikáciu s product ownermi.

2.1 Návrh UX obrazoviek

Pred implementáciou sme začali najprv návrhom UX obrazoviek. Tieto obrazovky majú výhodu, že je možné ich rýchlo nakresliť a pri nejakej nezhode s product ownermi sa dajú jednoducho prerobiť.

2.1.1 Obrazovka zobrazenie mapy s parkoviskami

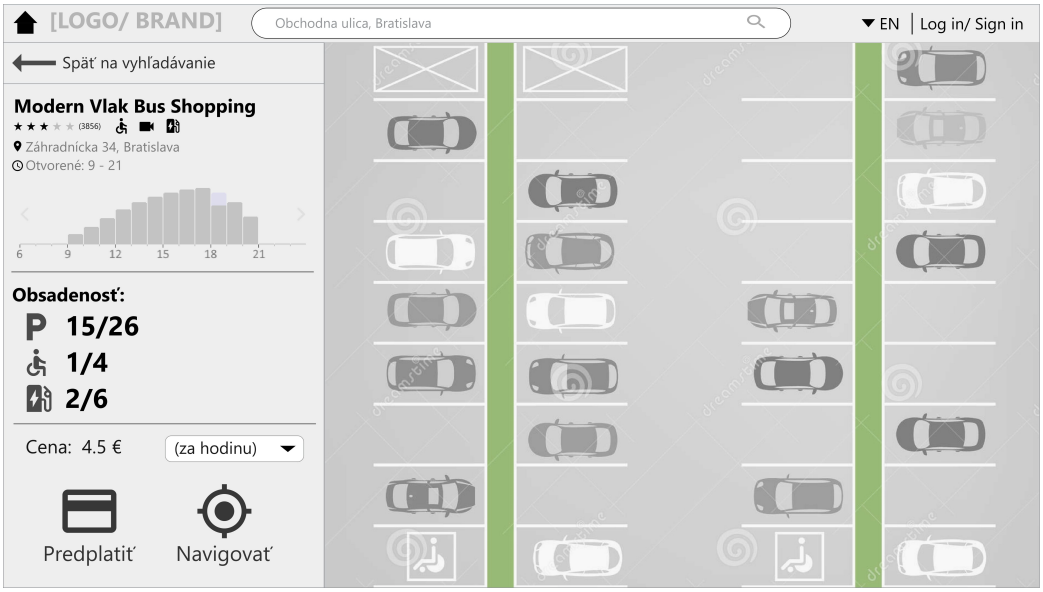
Na obrázku 2.1 môžeme vidieť návrh obrazovky mapy. Návrh obsahuje v hornej časti vyhľadávacie pole pre hľadanie mesta, mestskej časti alebo ulice. V ľavej časti môžeme vidieť panel s vyhľadávaným pojmom a zobrazené dostupné parkoviská zoradené podľa filtra, v tomto prípade od vzdialenosti od vyhľadávaného pojmu. V strede obrazovky sa nachádza mapa na ktorej sú zobrazene piny s parkoviskami. Príslušné písmeno značí, o ktoré parkovisko ide v ľavom panely.



Obr. 2.1: *UX obrazovka pre zobrazenie mapy*

2.1.2 Obrazovka zobrazenie detailu parkoviska

Na obrázku 2.2 môžeme vidieť návrh obrazovky pre zobrazenie detailu parkoviska. Návrh znova obsahuje v hornej časti vyhľadávacie pole. V ľavej časti sa nám zmenil obsah na informácie o konkrétnom parkovisku. Môžeme tu vidieť obsadenosť v čase, otváracie hodiny, aktuálnu obsadenosť... V strede sa nachádza vizualizácia konkrétneho parkoviska, ktorá nám zobrazuje voľné a obsadené parkovacie plochy.



Obr. 2.2: *UX obrazovka pre zobrazenie parkoviska*

2.2 Model údajov pre zobrazenie parkoviska

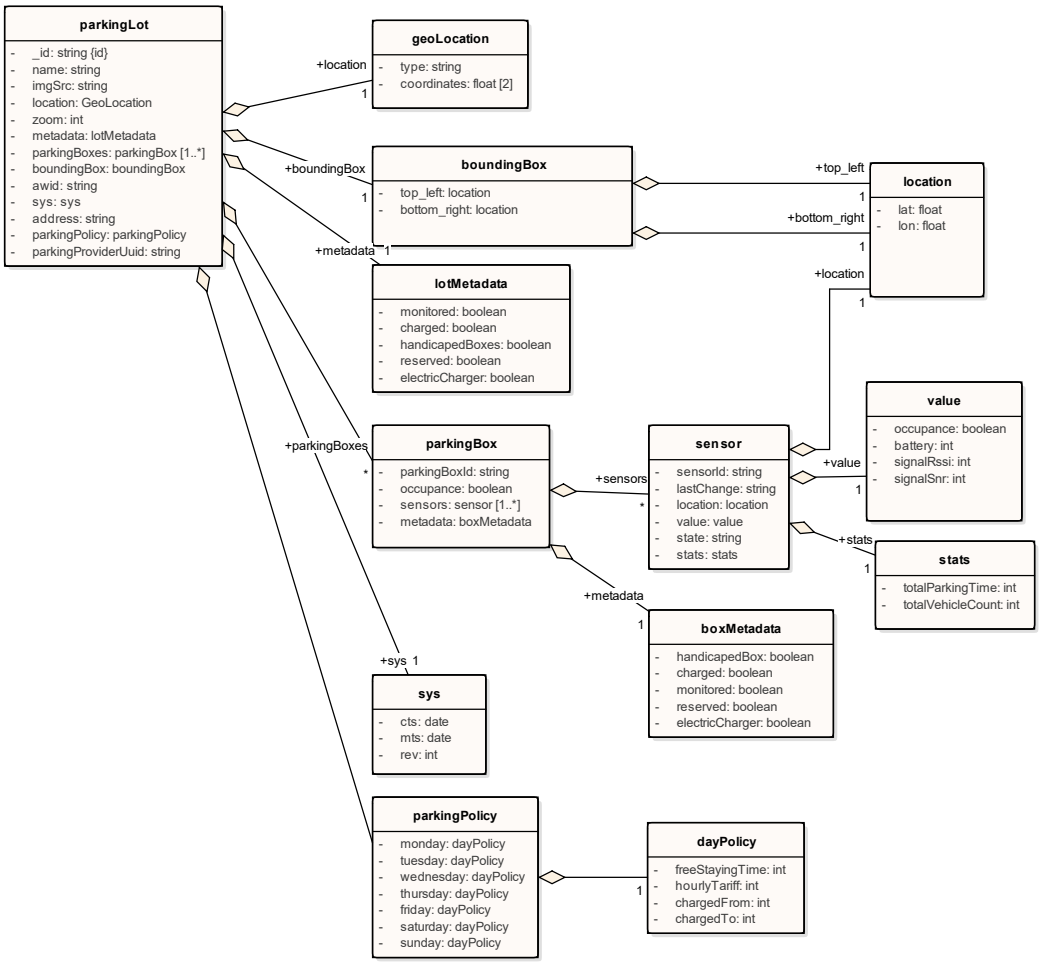
Pre ukladanie stavu parkovísk sa používa kolekcia *parkingLot*. Každý záznam parkoviska má nasledovné údaje (viď. obr.: 2.3):

- *name* - názov parkoviska
- *imgSrc* - referencia na obrázok parkoviska
- *location* - geojson pozície parkoviska
- *zoom* - atribút pre nastavenie vzdialenosti pohľadu na parkovisko
- *metadata* - doplnkové údaje o parkovisku:
  - *monitored* - či je parkovisko monitorované
  - *charged* - či má parkovisko špeciálne cenové politiky
  - *handicapedBoxes* - či sú miesta pre hendikepovaných
  - *reserved* - či sú rezervované miesta

  - *handicapedBoxes* - či sú miesta s elektricou prípojkou

- *parkingBoxes* - zoznam parkovacích miest s údajmi:
  - *parkingBoxId* - identifikátor parkovacieho miesta
  - *occupance* - či je miesto obsadené
  - *sensors* - hash mapa senzorov inštalovaných na parkovisku s údajmi:
    * *sensorId* - jednoznačný identifikátor senzora
    * *lastChange* - timestamp poslednej zmeny (odpovedá časovej známke poslednej prijatej správy)
    * *location* - geografická pozícia senzora
    * *value*
      · *occupance* - obsadenosť daného senzora
      · *battery* - stav batérie senzora
      · *signalRssi* - sila signálu senzora (received signal strength indicator)
      · *signalSnr* - pomer signálu k šumu (signal to noise ratio)
    * *state* - stav senzora (či je senzor aktívny, prípadne placeholder pre ďalšie dodatočne definované stavy)
    * *lastOccupanceChange* - timestamp poslednej zmeny obsadenosti senzora
    * *stats* - aktuálne štatistiky senzora
      · *totalParkingTime*
      · *totalVehicleCount*
      · *maxSignalRssi*
      · *minSignalRssi*
      · *maxSignalSnr*
      · *minSignalSnr*
      · *lastMessageId*
    * *created* - timestamp kedy bol senzor vytvorený
    * *verHW* - verzia hardvéru senzora

        * *verSW* - verzia softvéru (firmvéru) senzora
    - *metadata* - objekt pre doplnkové údaje o parkovacom mieste
    - *parkingPolicy* - pokiaľ má parkovacie miesto špeciálne cenové politiky, sú zadefinované pomocou tohto objektu
- *boundingBox* - geografické pozície pre umiestnenie obrázka parkoviska (ľavý horný a pravý dolný roh)
- *awid*
- *sys*
- *address*
- *parkingPolicy* - statické cenové politiky pre každý deň v týždni
    - monday
        * *freeStayingTime*
        * *hourlyTariff*
        * *chargedFrom*
        * *chargedTo*
- *parkingProvideUuid*

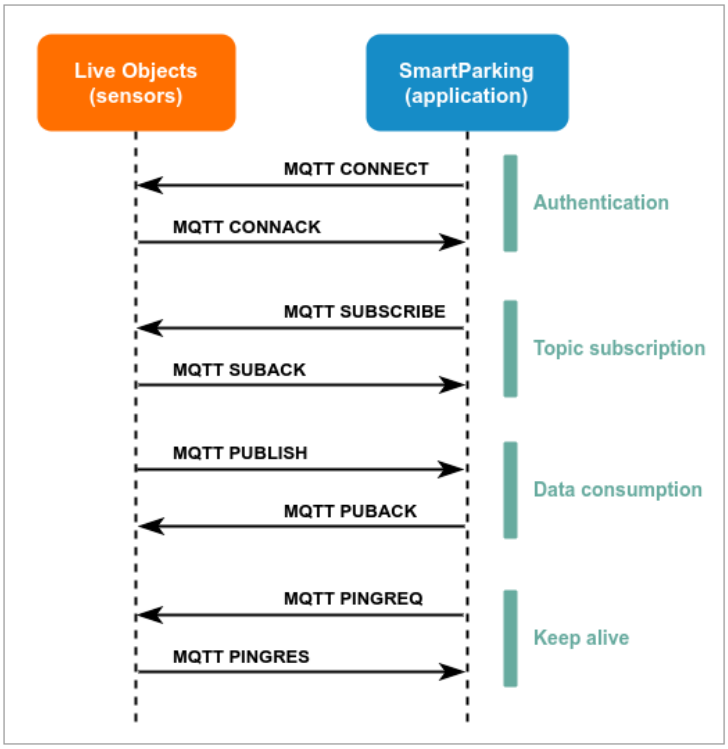


Obr. 2.3: *Diagram tried pre entitu parkingLot (parkovisko)*

3 Implementácia

3.1 Komunikácia s Live Objects

Komunikácia backendu so senzormi bola implementovaná použitím knižnice MQTT.js. Priebeh komunikácie medzi aplikáciou (klientom) a platformou Live Objects zobrazuje nasledovný diagram (Obr. 3.1):



Obr. 3.1: *Komunikácia s LO platformou*

Komunikácia bola v projekte implementovaná v rámci skriptu *mqtt/mqtt-client.js*. Tento skript exportuje metódu *connectAndSub()*, ktorá vytvára spojenie s Live Objects platformou a zabezpečuje prihlásenie sa na odber, a to vždy pri spustení aplikačného servera.

3.1.1 Konfigurácia komunikácie

Pri konfigurácii komunikácie je potrebné zabezpečiť, aby správne prebehla autentifikácia a prihlásenie sa na odber správ každého z tenantov pri spustení servera. Pre konfiguráciu komunikácie s Live Object platformou bolo vytvorené v rámci *development.json* konfiguračného súboru pole *subscriptions*. Štandardne má každý z tenantov zadefinovaný vlastný objekt pre subscription. Každý objekt v poli subscriptions obsahuje nasledovné parametre:

- *url* - jednoznačný identifikátor pre pripojenie k sprostredkovateľovi zariadení,
- *apiKey* - klúč pre autorizáciu,
- *topic* - topic pre odoberanie MQTT správ,
- *parkingProviderAwid* - AWID pre providera.

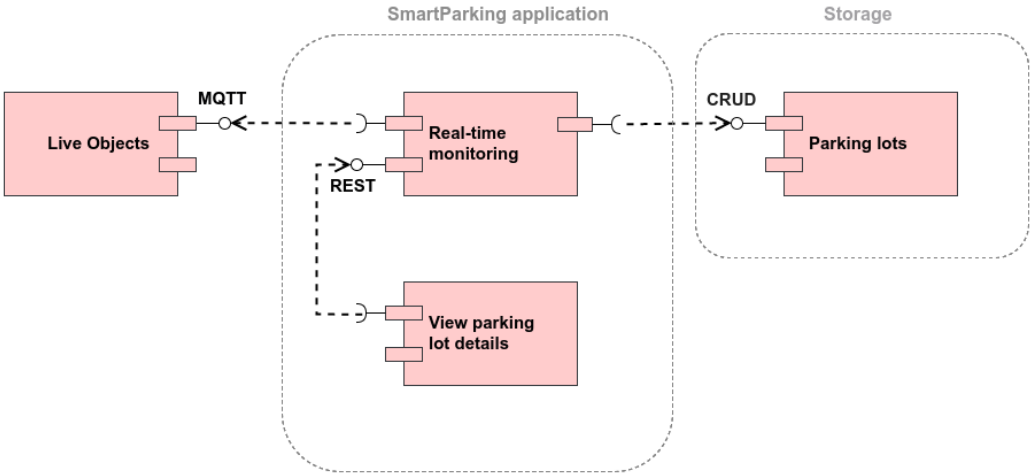
```
{
  "mqtt": {
    "subscriptions": [
      {
        "url": "mqtt://fut.m2m.orange.com:1883",
        "apiKey": "e25ab350fd044dffbfad3b56e702326f",
        "topic": "fifo/parkingsensor1",
        "parkingProviderAwid": "3b45e8465263439awed28e00dbcb5d8"
      },
      {
        "url": "https://liveobjects.orange-business.com",
        "apiKey": "d5dbb7abasdasdwa994656a92a6051",
        "topic": "fifo/e2e_platform_test",
        "parkingProviderAwid": "3b45e8465263439d898828e00dbcb5d8"
      },
    ]
  }
}
```

Ukážka 3.1: *Konfiguračný súbor development.json*

Metóda *connectAndSub()* sa následne postará o preiterovanie poľa subsrciptions a vytvorenie zadefinovaných pripojení.

3.1.2 Ukladanie stavu senzorov

Pre ukladanie prichádzajúcich zmien do MongoDB sme v *model/parking-lot-model* implementovali logiku, ktorá zabezpečuje komunikáciu s databázou. Pri každej novej zachytenej správe sa modul *mqtt* postará o zavolanie metóda *updateParkingLotSensor()*, ktorá si vyžaduje argumenty *awid* pre identifikáciu relevantného parkoviska a objekt *sensor*, ktorý obsahuje samotné dáta senzora. Celkový pohľad na architektúru využitú pri implementácií zobrazuje nasledovný diagram (Obr. 3.2):



Obr. 3.2: *Celkový náhľad na využitú architektúru*

Modul *mqtt* spracováva niekoľko typov správ:

- *BTT* - správa posielajúca informácie o stave batérie senzora z reálnych senzorov,
- *TMP* - správa posielajúca informácie o teplote senzora z reálnych senzorov,
- *SIM* - správa posielaná zo simulovaných senzorov.

Typ správy je pri každej správe identifikovaný pomocou poľa *messageType*. O extrakciu dát zo správ, dôležitých pre reprezentáciu senzorov v aplikácii sa

starajú metódy *handleMessageBTT()*, *handleMessageTMP()* a *handleMessageSIM()* v module *mqtt*. Extrahované dáta sú následne vstupným parametrom do spomínanej metódy *updateParkingLotSensor()* v podobe *sensor* objektu.

Každý z typov správa v sebe nesie vždy informáciu o stave obsadenosti daného senzora na parkovisku.

3.2 Vyhľadávanie parkovísk

Na vyhľadanie parkovísk sa použil komponent *<Select/>* z knižnice *react-select*. Na získanie miest záujmu podľa kľúčových slov, v ktorých okolí sa hľadajú parkoviská sa používa knižnica *Google Places*. Na získanie parkovísk v okruhu vybratého miesta záujmu sa používa rest API *getNearByParkingLots*.

3.2.1 Vyhľadávanie podľa miest záujmu

Keď je dĺžka textu v poli väčšia alebo rovná 1, tak sa pomocou knižnice Google Places odošle request na *nearbySearch* API. Obsahom requestu je základná alebo aktuálna poloha stredu mapy parkovísk, kľúčové slová použité na vyhľadávanie a ich jazyk. API vracia zoznam miest záujmu zoradených podľa vzdialenosti od zadaného stredu mapy, ktoré boli pomocou zadaných kľúčových slov nájdené. Názvy s adresami nájdených miest záujmu sa zobrazia pod textovým poľom. Po vybratí jedného z ponúknutých miest sa zobrazí mapa vycentrovaná na toto miesto, s parkoviskami, ktoré sa nachádzajú v okolí. Parkoviská v okolí sú vyhľadané v databáze pomocou API *getNearByParkingLots* a GPS súradníc vybratého miesta záujmu. V tomto volaní API sú vrátené všetky parkoviská z okruhu 50 km zadaných GPS súradníc.

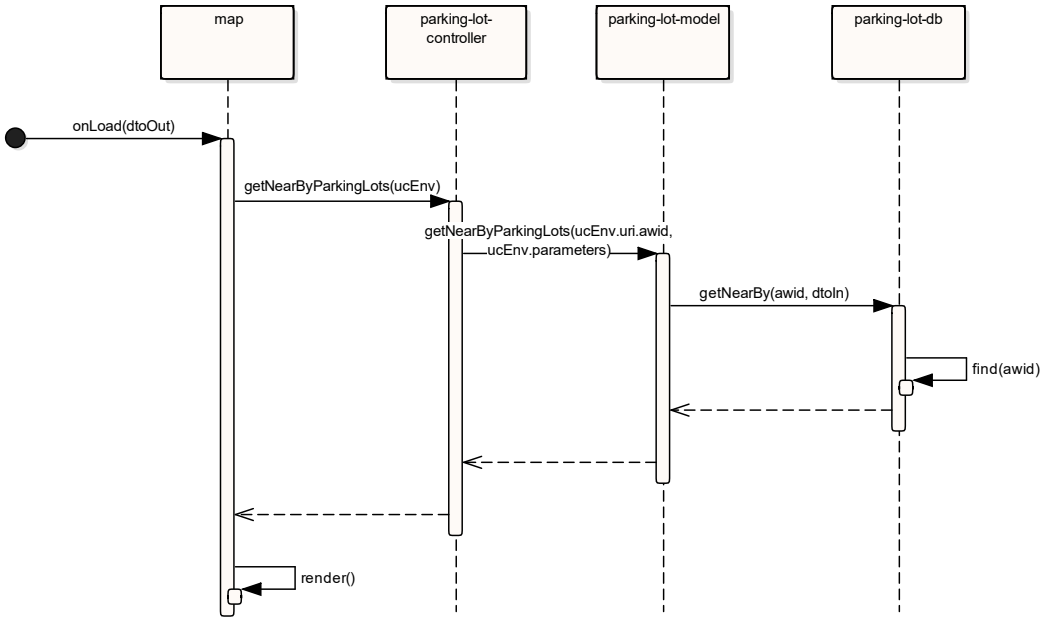
3.3 Vizualizácia mapy

Na mape sa používateľovi zobrazujú markery označujúce pozície parkovísk. Jednotlivé parkoviská si môže používateľ priblížiť kliknutím na príslušný marker.

Pre zobrazenie mapy boli implementované komponenty *<MapComponent/>* a *<GoogleMap/>*. Tieto komponenty taktiež zobrazujú markery na

pozíciách parkovísk.

Pri načítaní mapy sa odošle http request na API poskytujúcu funkciu *getNearByPakingLots()*, ktorá vráti údaje o parkoviskách z databázy. Z týchto údajov sa následne vyberú geografické pozície jednotlivých parkovísk a na príslušných pozíciách na mape sa zobrazia markery s odkazmi na zobrazenie konkrétnych parkovísk. (Viď Obr. 3.3)



Obr. 3.3: *Sekvenčný diagram pre zobrazenie mapy*

3.4 Vizualizácia parkoviska

Po vybratí parkoviska sa zobrazí mapa s obrázkom parkoviska a indikátormi obsadenosti pre každé parkovacie miesto.

Pre zobrazenie vybraného parkoviska sa okrem komponentov *<MapComponent/>* a *<GoogleMap/>* používa aj komponent *<GroundOverlay/>* zodpovedný za zobrazenie obrázka parkoviska na mape. Taktiež sa na obrázku zobrazujú indikátory obsadenosti (zelené a červené body) pre každé parkovacie miesto.

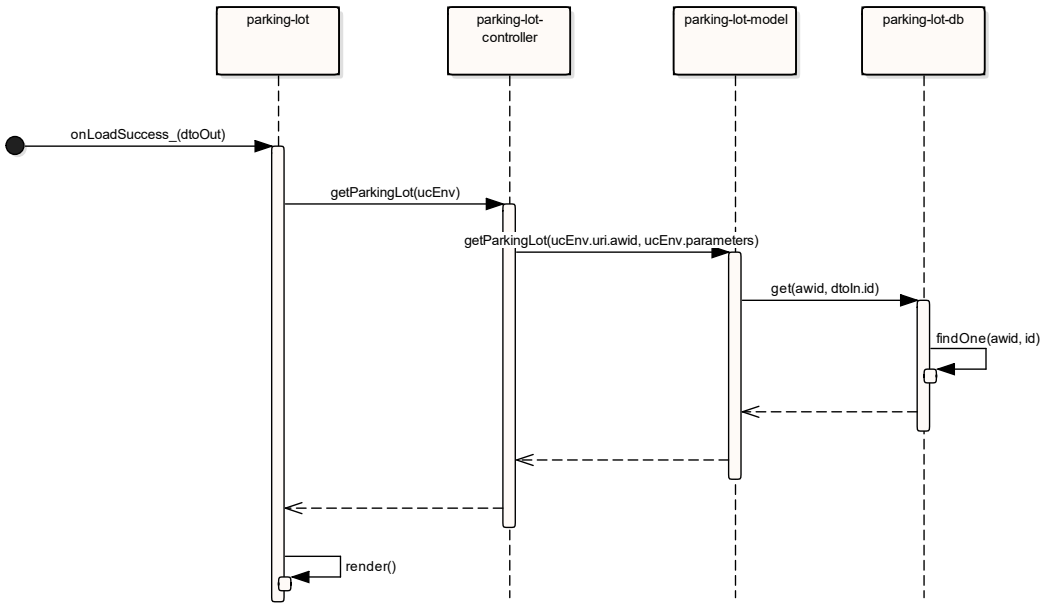
Po kliknutí na marker označujúci parkovisko sa otvorí nové okno, ktoré

pri načítaní odošle 2 http requesty:

- getParkingLot(id)
- getImage(id)

API poskytujúca funkciu *getParkingLot(id)* vráti údaje o parkovisku, medzi ktorými je pole senzorov parkoviska s pozíciami senzorov (*parkingLot.parkingBoxes[i].sensors[j].location)* a obsadenosťou senzorov (*parkingLot.parkingBoxes[i].sensors[j].value.occupance*). Na základe týchto atribútov sú na obrázku zobrazené indikátory (occupance == true červený, occupance == false zelený bod).

API poskytujúca funkciu *getImage(id)* vráti obrázok vo formáte *base64*. Takýto obrázok sa transformuje na formát *png* a zobrazí sa na mape. (Viď Obr. 3.4)



Obr. 3.4: *Sekvenčný diagram pre zobrazenie parkoviska*

3.5 Implementácia obrazoviek

Pri implementácií obrazoviek sme sa snažili vychádzať čo najviac z navrhnutých obrazoviek. Do určitej miery sa nám to podarilo, avšak je potrebné si

uvedomiť, že ide len o prvotné implementácie a v neskorších iteráciách sa obrazovky budú meniť.

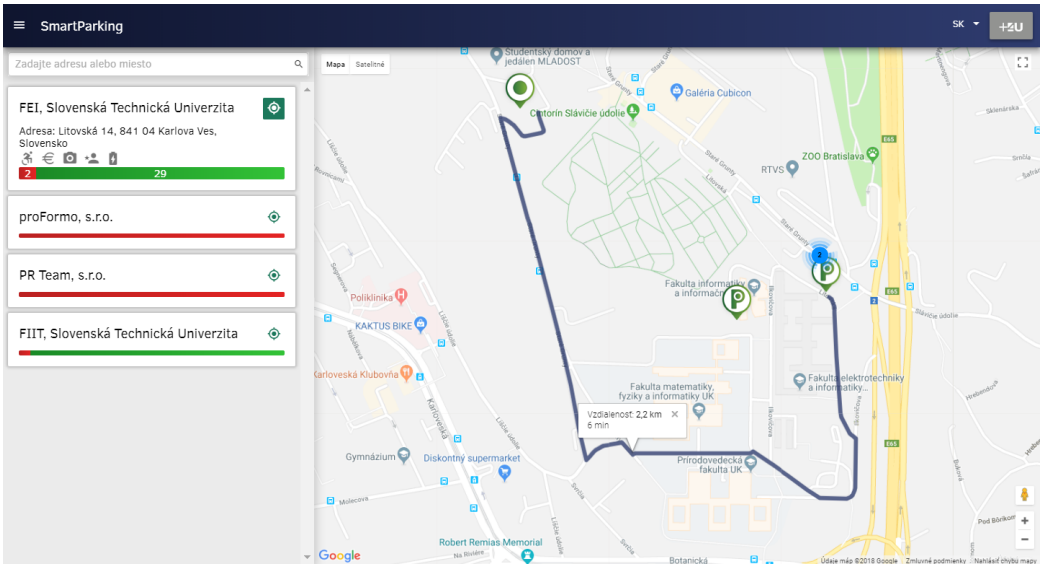
3.5.1 Implementácia domovskej stránky

Na obrázku 3.5 môžeme vidieť implementáciu domovskej stránky. Na tejto stránke je možné vyhľadať parkoviská v zadanej oblasti.

Obr. 3.5: *Implementácia domovskej stránky*

3.5.2 Implementácia obrazovky zobrazenie mapy s vyhľadanými parkoviskami

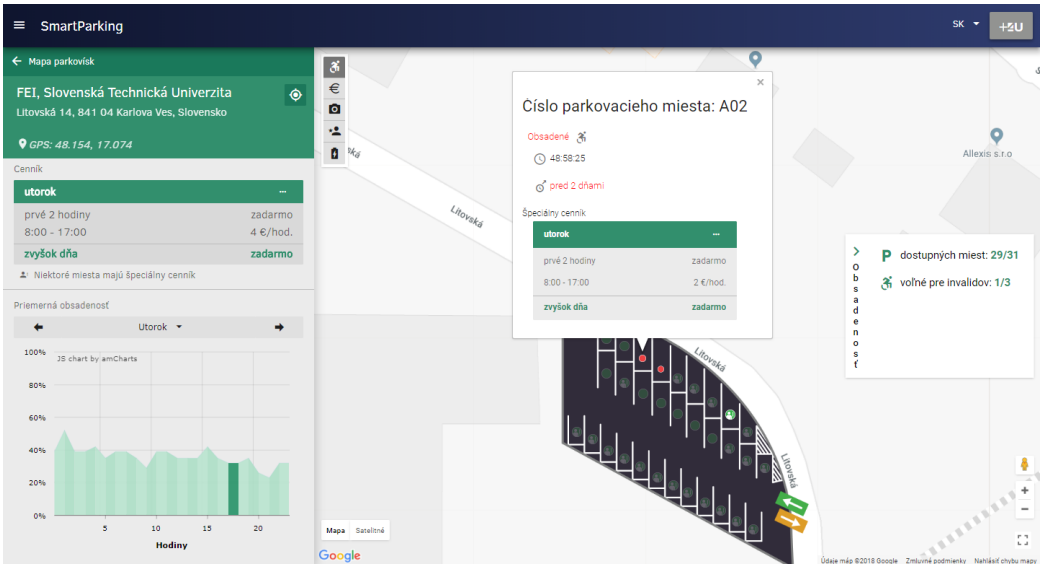
Na obrázku 3.6 môžeme vidieť implementáciu obrazovky zobrazenie mapy s vyhľadanými parkoviskami. V ľavom paneli je vyhľadávacie pole. Pod vyhľadávacím poľom sa nachádza zoznam viditeľných parkovísk na mape. V strede vidíme zobrazenú mapu na ktorej sú rozmiestnené markery s lokalitami parkovísk. Po kliknutí na marker, alebo na položku zoznamu v ľavom paneli sa nám zobrazí detail konkrétneho parkoviska. Na obrázku 3.6 taktiež vidieť implementovanie zhlukovania markerov. Okrem toho je ponúkaná možnosť zobrazenia najkratšej cesty k vybranému parkovisku z aktuálnej polohy.



Obr. 3.6: *Implementácia obrazovky zobrazenie mapy*

3.5.3 Implementácia obrazovky zobrazenie parkoviska

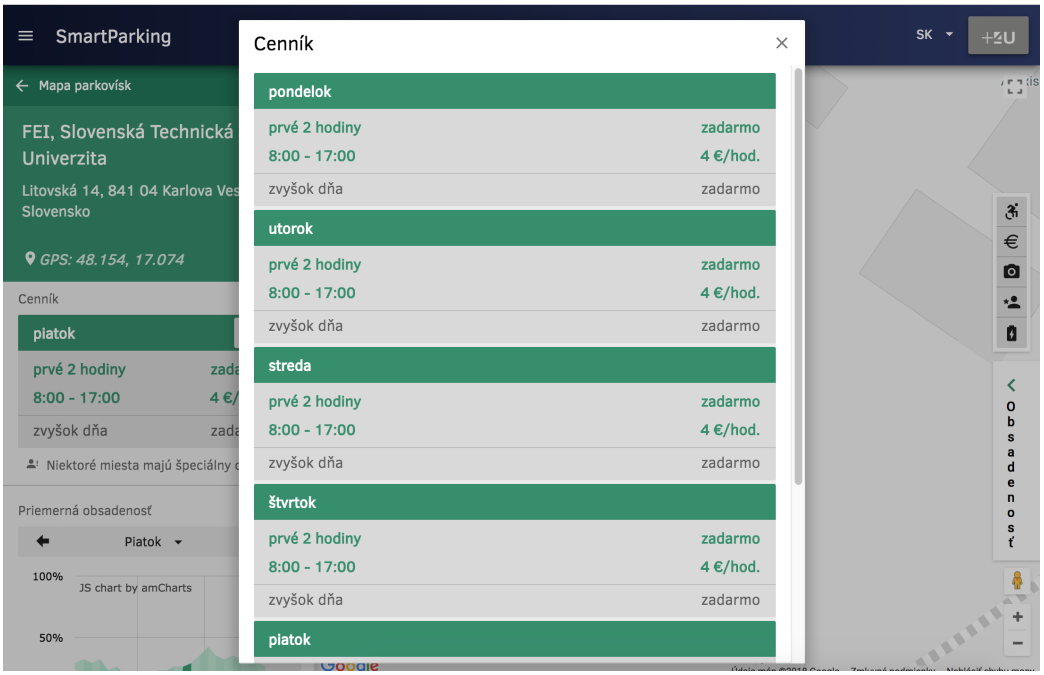
Na obrázku 3.7 môžeme vidieť implementáciu obrazovky zobrazenie parkoviska. V ľavom paneli sa nachádzajú aktuálne informácie o obsadenosti a štatistiky obsadenosti parkoviska. Môžeme vidieť, že sa do určitej miery zhoduje s návrhom na obrázku 2.2. V strede máme zobrazenie konkrétneho parkoviska. Zelené body označujú parkovacie miesta, ktoré sú voľné, červené body parkovacie miesta, ktoré sú obsadené, a body s ikonou poukazujú na dodatkové informácie o parkovacom mieste. Po rozkliknutí identifikátorov obsadenosti sa zobrazia informácie príslušného parkovacieho miesta.



Obr. 3.7: *Implementácia obrazovky zobrazenie parkoviska*

3.5.4 Implementácia komponentu cenových politík

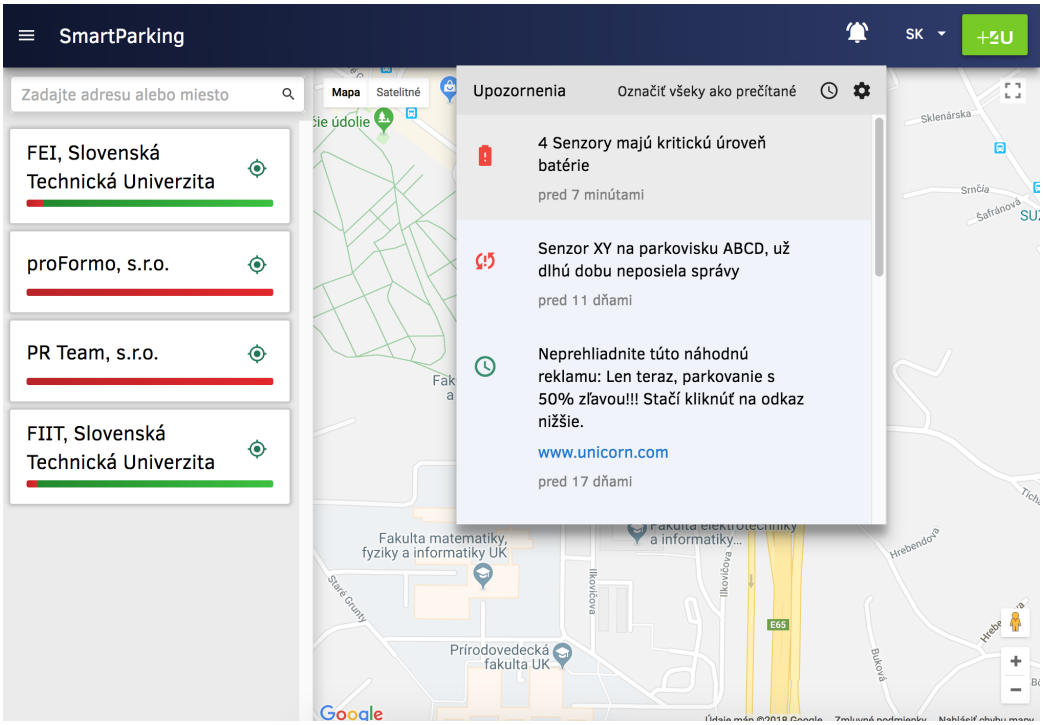
Komponent bol navrhnutý aby bol zrozumiteľný, malý a podporoval znovupoužiteľnosť. Komponent má hlavičku v ktorej zobrazuje deň, pre ktorí platí cenová politika a tlačidlo umožňujúce zobraziť prehľad cenových politík na celý týždeň. V tele komponentu sú zobrazené detaily cenovej politiky ako *doba parkovania zadarmo*, *spoplatnená doba parkovania* a *hodinová tarifikácia*. Telo komponentu sa mení a nezobrazuje nepotrebné informácie. Taktiež komponent zvýrazňuje práve platnú tarifikáciu. Na obrázku 3.7 môžme vidieť, že parkovanie je zadarmo, keďže snímok bol vytvorený mimo spoplatnených hodín. Na obrázku 3.8 môžeme vidieť implementáciu obrazovky prehľad parkovacích politík. Obrazovka sa zobrazí po kliknutí na ikonu troch bodiek komponentu parkovacích politík.



Obr. 3.8: *Implementácia obrazovky prehľad parkovacích politík*

3.5.5 Implementácia komponentu panel upozornení

Komponent panelu upozornení bol navrhnutý pre prihlásených používateľov, aby poskytol jednoduchú možnosť upozorniť na dôležité udalosti v systéme. Samotný panel aj so vzorovými upozorneniami môžme vidieť na obrázku 3.9. V hlavičke komponentu máme skupinu tlačidiel na správu upozornení ako *Označiť všetky ako prečítané*, *Všetky upozornenia* a *Nastavenia*. V tele komponentu je zoznam najnovších upozornení, pre ktoré bol vytvorený samostatný komponent. Každé upozornenie nesie v sebe informáciu o svojej *závažnosti*, čomu zodpovedá zafarbenie jeho ikony, *ikonu*, *text upozornenia*, *dátum* kedy upozornenie vzniklo a podporuje tiež vnorený obsah, čo umožní upozornenia dodatočne prispôsobiť.



Obr. 3.9: *Implementácia komponentu panel upozornení*

4 Testovanie

Testovanie komunikácie s LO platformou bolo vykonané pomocou simulátora pre posielanie správ. Boli zadefinované rôzne typy správ, pričom simulátor umožňuje simulovať ich posielanie na poskytovateľa, zadefinovaného pomocou URL a na konkrétny *topic* v rámci tohto poskytovateľa. Konfigurácia simulátora bola spravidla totožná s konfiguráciou skutočného poskytovateľa správ, pričom formát správy bol tiež totožný s formátom správy z reálnych senzorov. Poskytovateľ (LO) sa následne postará o vypublikovanie týchto správ na príslušný topic, na ktorý bola naša klientská aplikácia v momente testovania prihlásená pre odber.