FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Zobrazení dat o průjezdu vozidel

KIV/ASWI - Specifikace požadavků

Plzeň, březen 2018

David Pivovar
pivovar@students.zcu.cz
Jan Kohlíček
kohl@students.zcu.cz
Michal Horký
horkmi@students.zcu.cz
Zdeněk Valeš
valesz@students.zcu.cz

1 Obsah

1 Obsah **1**

2 Úvod **2**

2.1 Případy užití 2

2.2 Návrhy do budoucna 3

3 Architektura **4**

4 Server **5**

4.1 Databáze 5

4.1.1 Schéma databáze 5

4.2 Automatické načítání dopravních dat 5

4.2.1 Open data 5

4.3 Rest API 8

4.3.1 Verzování 8

4.3.2 Formát dat 8

4.3.3 Stavové kódy HTTP 8

4.3.4 Autentizace 8

4.3.5 URI 9

4.3.6 Testování 10

4.4 Cron 10

4.5 Autentizace 10

5 Webová aplikace **11**

5.1 Uživatelské rozhraní 11

5.2 Vizualizace 11

5.3 Návrh UI 12

6 Technologie **14**

7 Rizika a omezení **15**

8 Testování **16**

2 Úvod

V současné době neexistuje systém, který by dokázal efektivně zpracovat data týkající se dopravy získaná měřením v různých částech Plzeňského kraje. Pravidelní i příležitostní řidiči tak nemají rychlý přehled o současné či dlouhodobé dopravní situaci, kterému by eventuálně mohli přizpůsobit svoji jízdu. Správa Plzeňského kraje pak nemůže tato data využít k efektivnímu plánování uzavírek během případných oprav komunikací.
Cílem projektu je vytvoření systému, který bude schopen existující data automaticky načítat a následně umožní jejich volné zobrazení na webu s využitím mapových podkladů, včetně vizualizace vybraných statistických údajů.

2.1 Případy užití

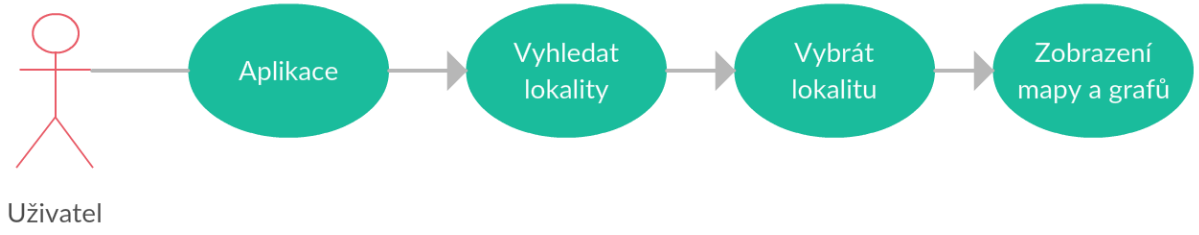
Řidiči:

- Zobrazení statistických dat o hustotě dopravy v danou hodinu
- Zobrazení mapy s vyznačenou mírou dopravy v jednotlivých oblastech
- Plánování optimální trasy

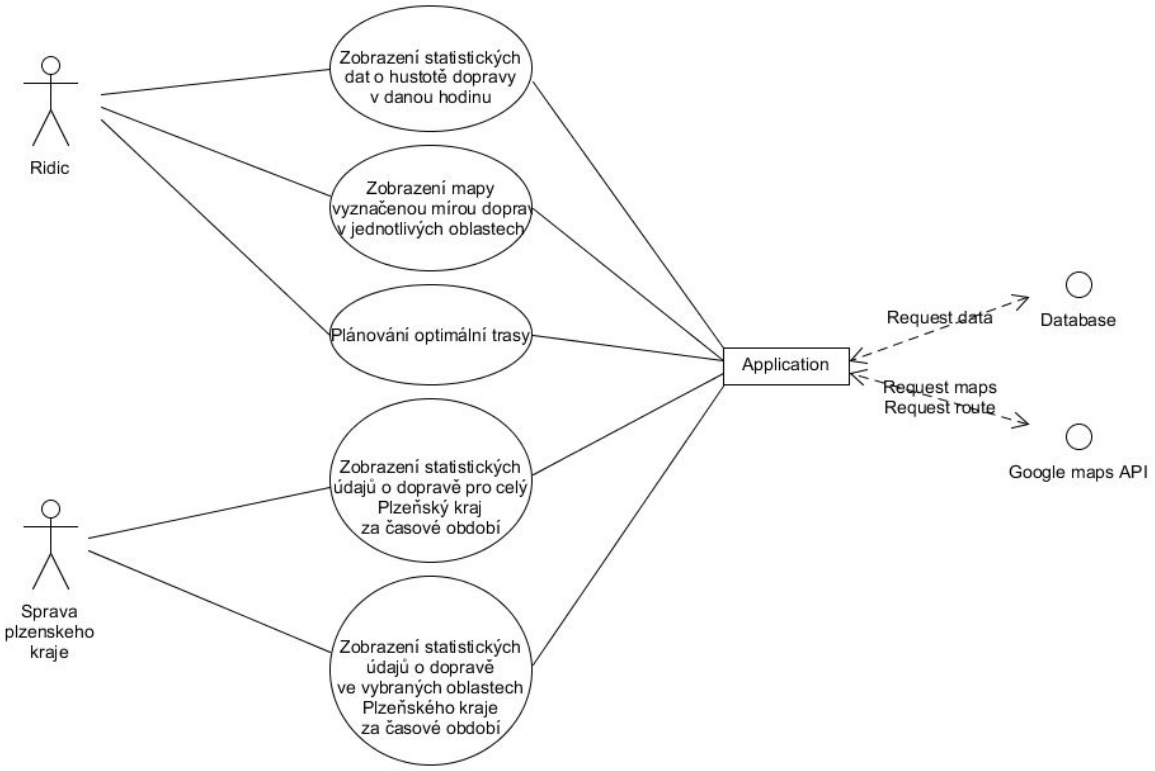
Správa plzeňského kraje:

- Zobrazení statistických údajů o dopravě pro celý Plzeňský kraj za časové období
- Zobrazení statistických údajů o dopravě ve vybraných oblastech Plzeňského kraje za časové období
- Plánování oprav komunikací a optimálních objízdných tras

Práce s aplikací bude velmi jednoduchá, jak je vidět na use-case diagramu:



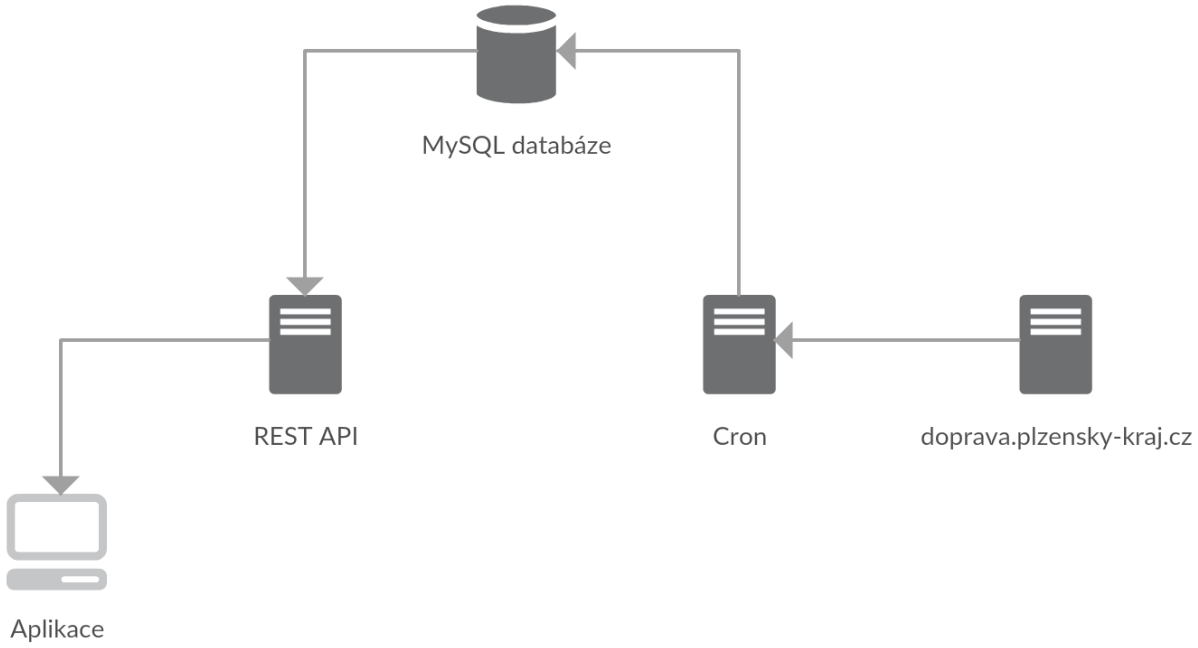
Use-case diagram funkci systemu:



2.2 Návrhy do budoucna

- Využití Google API pro plánování tras.

3 Architektura



4 Server

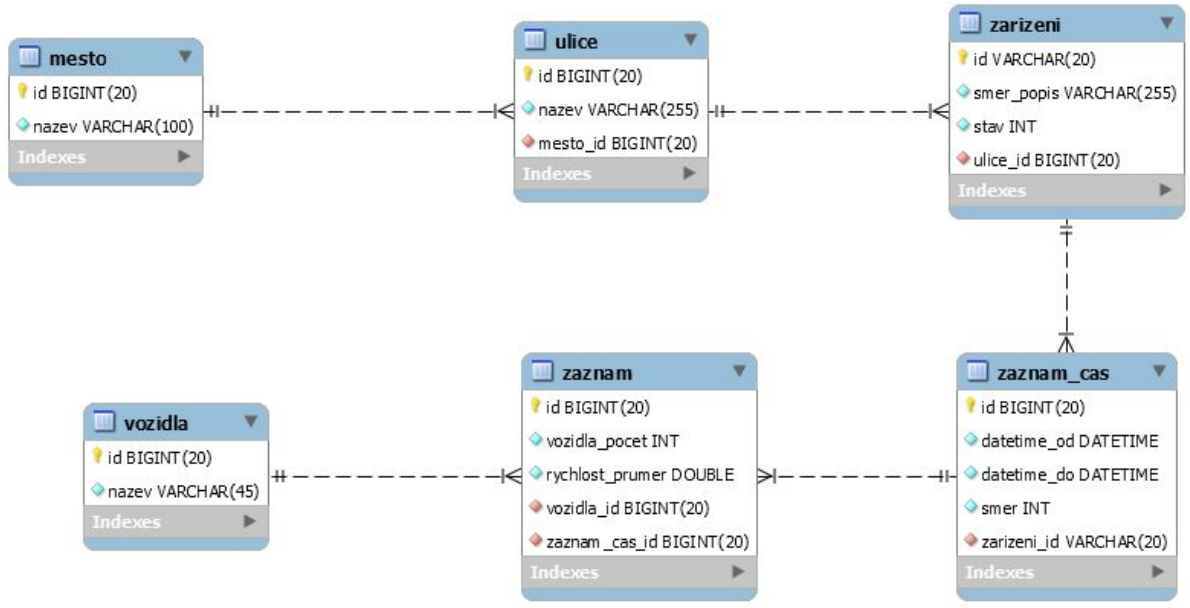
Serverová část aplikace se bude skládat ze dvou částí. První část bude periodicky (jednou-dvakrát za den) načítat data ze stránek Plzeňského kraje a ukládat je do databáze. Druhá část bude API, skrze které bude možné data z databáze získat. Primární účel druhé části je poskytování dat do uživatelského rozhraní.

4.1 Databáze

Pro uložení dat bude využita databáze MySQL. Databáze bude uchovávat data za poslední rok (případně půl roku, záleží na celkové velikosti dat). Starší data se budou průběžně automaticky mazat.

4.1.1 Schéma databáze

Databáze bude obsahovat tabulku s lokacemi, tabulku se statistickými údaji k jednotlivým snímačům za hodinu a tabulku se statistickými údaji všech snímačů za časovou jednotku (hodinu).



4.2 Automatické načítání dopravních dat

Každý den se ze stránek plzeňského stáhnou data pro předchozí den. Data ve formátu zip se rozbalí a nahrají do databáze.

4.2.1 Open data

Plzeňský kraj zpřístupňuje data na těchto adresách:
http://doprava.plzensky-kraj.cz/opendata/doprava/rok/

http://doprava.plzensky-kraj.cz/opendata/doprava/tyden/
http://doprava.plzensky-kraj.cz/opendata/doprava/den/

Názvy souborů jsou přesně definovány pro rok *“DOPR_R_RRRR.zip”*, týden *“DOPR_T_RRRRTT.zip”* a den *“DOPR_D_RRRRMMDD.zip”*. Kde *RRRR* je rok, *TT* týden, *MM* měsíc a *DD* den v měsíci.

např.: http://doprava.plzensky-kraj.cz/opendata/doprava/den/DOPR_D_20180309.zip

Bude se pracovat jen s daty pro daný (uplynulý) den. Soubor zip obsahuje **tři** soubory *DOPR_D_RRRRMMDD.csv*, *Locations.csv* a *ReadME.txti*, struktura je popsána v následující tabulce.

Název sloupce	**Popis**
idDetektor	Formát 10<IdZarizeni>10<smer>; např. 10105102
DatumCas	Datum a čas pořízení záznamu; např. "2018-03-09 00:00:00.946".
Intenzita	Počet vozidel v době detekce. Předpokládá se 1 vozidlo u jednotlivých detekcí.
IntenzitaN	Přepočtená intenzita vozidla s ohledem na velikost vozidla.
Obsazenost	Procentuální obsazenost detektoru.
Rychlost	Rychlost vozidla v km/h.
Stav	Stav detektoru: 1 - funkční detektor, <0 nefunkční detektor.
TypVozidla	Jednoduché označení typu vozidla: 1 - bike, 2 - car, 3 - van, 4 - truck.
Trvani100	Časový rozestup mezi daty v historii ve stovkách milisekund.
RychlostHistorie	Historie rychlostí k dané detekci vozidla.
TypVozidla10	Plné označení typu vozidla: 0 - unknown, 1 - motorbike, 2 - car, 3 - car with trailer, 4 - van, 5 - van with trailer, 6 - light truck, 7 - light truck with trailer, 8 - truck, 9 - trick with trailer, 10 - bus.

Jednoznačný identifikátor záznamu je kombinace *IdDetektor* a *DatumCas*. <IdZarizeni> je vždy trojmístné číslo, <smer> může nabývat hodnotou 1 - ve směru nebo 2 - v protisměru. Struktura souboru Location.csv je popsána v následující tabluce.

Název sloupce	**Popis**
Name	Popis umístění.
Town	Název města.
Street	Název ulice.
IdDevice	Identifikátor měřícího zařízení ve tvaru KP<IdZarizeni>; např KP055.
idArea	Číslo, které udává skupinu – teritorium, ve které je několik zařízení. Např. Klatovy, Domažlice …

Jednoznačný identifikátor záznamu je *IdDevice*.

4.3 Rest API

Komunikace uživatelského rozhraní se serverem bude probíhat skrze REST API, které bude realizováno v jazyce PHP s použitím frameworku Lumen.

4.3.1 Verzování

Verze API bude v URL např. /api/v1/

4.3.2 Formát dat

Data budou k dispozici jen ve formátu JSON. Typ XML nebude podporován.

4.3.3 Stavové kódy HTTP

V REST API se budou využívat jen tyto stavové kódy:

- **200 OK** - Zpracování požadavku proběhlo bez problému.
- **400 Bad Request** - Požadavek nebyl zpracován. Důvodem může být serverem neznámý požadavek nebo špatná vstupní data.
- **401 Unauthorized** - Požadavek nebyl zpracován. Žadatel nebyl autorizován.
- **404 Not Found** - Požadovaný záznam nebyl nalezen.
- **500 Internal Server Error** - Požadavek nebyl zpracován. Nastala nečekaná chyba.

4.3.4 Autentizace

V HTTP hlavičce se bude nacházet token: ***Authorization: Bearer <token>***
Token bude typu **JWT**, odkaz na knihovnu: https://github.com/firebase/php-jwt
Pokud token v hlavičce nebude nebo nepůjde dekódovat odešle se **401 Unauthorized**.
Token bude obsahovat: **IP**, **prohlížeč** a **expiraci**, pokud se nebude obsah shodovat s odesilatelem požadavku, tak byl token odcizen a odešle se **401 Unauthorized**.
Při prošlé expiraci se také odešle **401 Unauthorized**.

4.3.5 URI

GET *api/v1/devices*
HEADER: ***Authorization: Bearer <token>***
PARAMS: (string) **address**
(bool) **showDirection**

příklad vrácených dat:
[
  {
    id: <zarizeni.id>,
    street: <ulice.nazev>
    town: <mesto.nazev>
    name: <zarizeni.smer_popis>
    direction: <smer> - odesílat jen pokud je **showDirection** == 1
  }
]

GET *api/v1/devices/<ID_Zarizeni>*
HEADER: ***Authorization: Bearer <token>***
PARAMS: (date) **dateFrom** - NULL - znamená poslední vložený den
(date) **dateTo** - NULL - znamená poslední vložený den
(time) **timeFrom** - NULL - znamená 00:00
(time) **timeTo** - NULL - znamená 23:59
(int) **direction** - NULL - znamená oba směry, 1 - ve směru, 2 - protisměru

příklad vrácených dat:
{
  id: <zarizeni.id>,
  street: <ulice.nazev>
  town: <mesto.nazev>
  name: <zarizeni.smer_popis>
  dateFrom:
  dateTo:
  direction: <**direction**> - může se zkopírovat hodnota z **direction**
  traffics: [
    {
      timeFrom:
      timeTo:
      speedAverage:
      typeVehicle:
      numberVehicle:
    }
  ]
}

GET ***api/v1/devices/<ID_Zarizeni>/time-period***
HEADER: ***Authorization: Bearer <token>***
PARAMS: (date) **dateFrom** - NULL - znamená poslední vložený den
(date) **dateTo** - NULL - znamená poslední vložený den
(time) **timeFrom** - NULL - znamená 00:00
(time) **timeTo** - NULL - znamená 23:59
(int) **direction** - NULL - znamená oba směry, 1 - ve směru, 2 - protisměru

POPIS: pole **trafics** bude obsahovat záznamy, které splňují tyto podmínky:

- zarizeni.id = ***<ID_Zarizeni>***
- **dateFrom** <= DATE(zaznam_cas.datetime_od)
- DATE(zaznam_cas.datetime_do) <= **dateTo**
- **timeFrom** <= TIME(zaznam_cas.datetime_od)
- TIME(zaznam_cas.datetime_do) <= **timeTo**
- **direction** IS NULL OR **direction** = zaznam_cas.smer

Poté se záznamy **seskupí** podle **timeFrom**, **timeTo** a **typeVehicle**.
speedAverage bude obsahovat průměr rychlostí seskupených záznamů a **numberVehicle** je počet záznamů, které se seskupily.

příklad vrácených dat:

```
{
  id: <zarizeni.id>,
  street: <ulice.nazev>
  town: <mesto.nazev>
  name: <zarizeni.smer_popis>
  dateFrom: 2018-01-01
  dateTo: 2018-01-30
  direction: <direction> - může se zkopírovat hodnota z direction
  traffics: [
    {
      timeFrom: 00:00
      timeTo: 00:15
      speedAverage: 60
      typeVehicle: 1
      numberVehicle: 21343
    },
    {
      timeFrom: 00:00
      timeTo: 00:15
      speedAverage: 70
      typeVehicle: 2
      numberVehicle: 31242
    },
```

```
    {
      timeFrom: 00:15
      timeTo: 00:30
      speedAverage: 50
      typeVehicle: 1
      numberVehicle: 12456
    },.....
  ]
}
```

GET ***api/v1/devices/<ID_Zarizeni>/day-period***
HEADER: ***Authorization: Bearer <token>***
PARAMS: (date) **dateFrom** - NULL - znamená poslední vložený den
(date) **dateTo** - NULL - znamená poslední vložený den
(int) **direction** - NULL - znamená oba směry, 1 - ve směru, 2 - protisměru

POPIS: data budou seskupena podle datum dne, vozidlo_id

příklad vrácených dat:

```
{
  id: <zarizeni.id>,
  street: <ulice.nazev>
  town: <mesto.nazev>
  name: <zarizeni.smer_popis>
  dateFrom: hodna z parametru dateFrom
  dateTo: hodna z parametru dateTo
  direction: <direction> - hodnota z parametru direction
  traffics: [
    {
      date: <datum dne ve formatu ‘yyyy-MM-dd’>,
      speedAverage: <zaznam_prum_den.rychlost_prumer>,
      numberVehicle: <zaznam_prum_den.vozidla_pocet>,
      typeVehicle: <zaznam_prum_den.vozidlo_id>
    },
    {
      date: <datum dne ve formatu ‘yyyy-MM-dd’>,
      speedAverage: <zaznam_prum_den.rychlost_prumer>,
      numberVehicle: <zaznam_prum_den.vozidla_pocet>,
      typeVehicle: <zaznam_prum_den.vozidlo_id>
    }
  ]
}
```

GET *api/v1/vehicles*
HEADER: ***Authorization: Bearer <token>***

příklad vrácených dat:

```
[
  {
    id: <vozidla.id>,
    name: <vozidla.nazev>
  }
]
```

4.3.6 Testování

Testování proběhne pomocí aplikace Postman.

4.4 Cron

Automatické spuštění scriptu pro stažené a nahraní dat do databáze zajistí **CRON** script, který bude napsán v **PHP**.

4.5 Autentizace

Vzhledem k povaze aplikace nebude vyžadována jakákoliv autentizace a aplikace tak bude volně přístupná.

5 Webová aplikace

Aplikace na základě získaných dat umožní vizualizaci průjezdnosti jednotlivých tras v Plzni. Zároveň bude možné zobrazit různé statistické údaje (průměrné vytížení, graf vytíženosti přes den) u jednotlivých tras, ulic a oblastí.

Uživatelské rozhraní bude navrženo tak, aby bylo aplikaci možné ovládat i z mobilního zařízení.

5.1 Uživatelské rozhraní

1. Nalezení lokality s umístěným radarem
    - zobrazení radarů na mapě
    - vyhledávání podle adresy (zobrazení vice radarů v blízkosti)
2. Zobrazení grafů o počtu vozidel, průměrné rychlosti, typu vozidel za časový interval
3. Plánovaní ideální trasy
    - zobrazení radarů na trase
    - zobrazení průmerných dat z celé trasy nebo zvoleného radaru

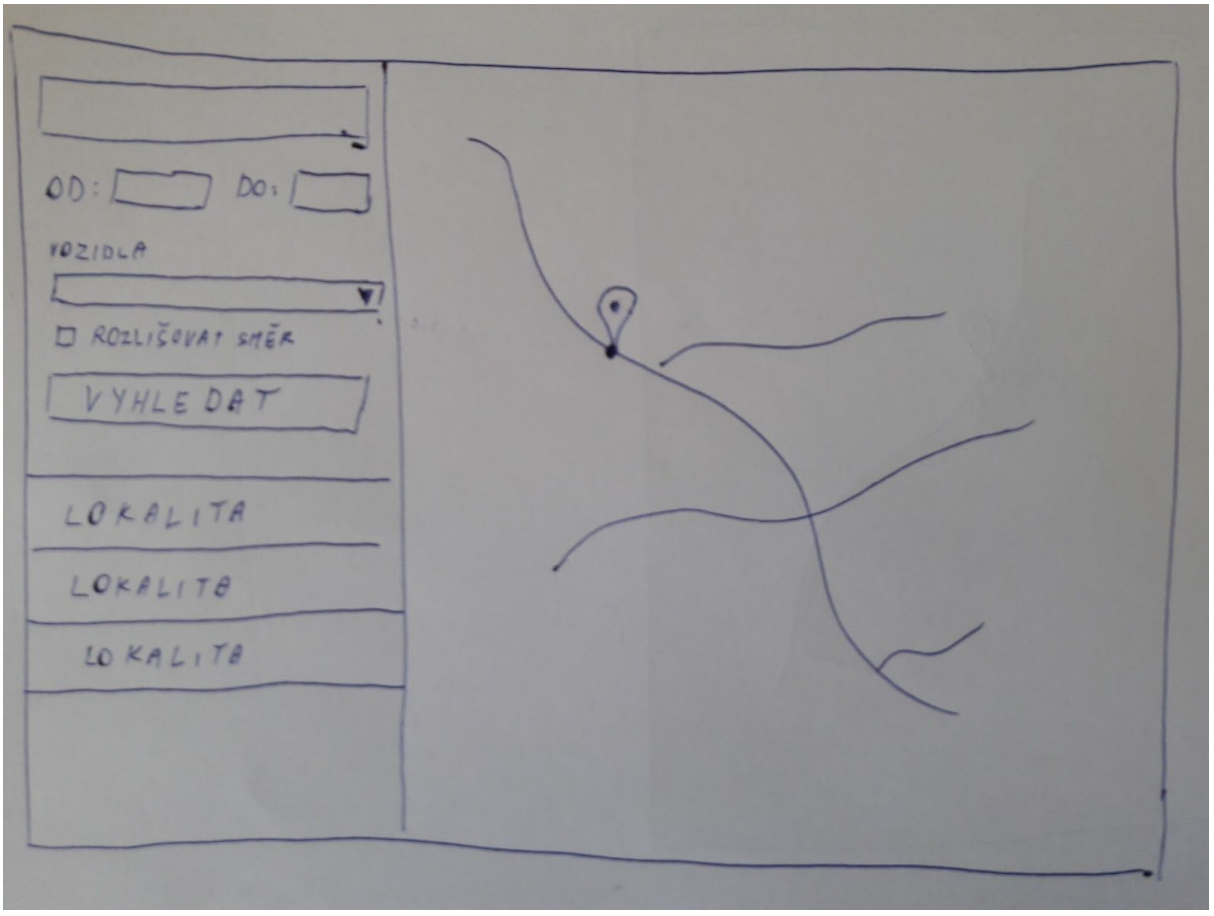
5.2 Vizualizace

K vizualizaci dopravní vytíženosti v jednotlivých oblastech bude využita technologie Google Maps, která v neplaceném režimu umožňuje 25 000 zobrazení map denně. V případě překročení tohoto limitu je použít služby zpoplatněno $0.50 za každých dalších 1 000 zobrazení navíc. U Android a iOS zařízení je počet zobrazení zdarma neomezený.
Pokud by počet zobrazení zdarma byl nedostatečný a byla striktně vyžadována neplacená technologie, bude použito OpenLayers.
Úroveň zobrazených detailů se bude odvíjet od velikosti zobrazované oblasti. Paleta barev použitá k zobrazení dopravní vytíženosti bude upřesněna během projektu.

UI bude vytvořeno ve frameworku AngularJS.

5.3 Návrh UI

První návrh:



Druhý návrh:

6 Technologie

Veskere pouzite technologie musi byt spustitelne na skolnim prostredi (pripadne na zakladni verzi placeneho webhostingu) bez nutnosti instalovat dalsi technologie.

- HTTP/CSS
- PHP 5
- Bootstrap
- Lumen
- Google maps API

Psani a komentar kodu dle programatorskych konvenci.

7 Rizika a omezení

Data z radarů jsou rozsáhlá (cca 1.5 milionu záznamů za den). Data se proto musí zredukovat (např. průmery za hodinu). Pro vyhledavani velkych casovych intervalu (napr. 1 rok) budou v databazi paralelne data zprumerovana za cely den.

Google maps umožňují 25 000 bezplatných zobrazení denně. Při překročení 25 000 zobrazení se za každé další zobrazení účtuje 0.50$. V případě, že by počet zobrazení přesáhl 25 000 (což se nepředpokládá) a nebyly by finanční prostředky pro financování Google maps, muselo by se pro zobrazování mapových podkladů použít jiné API.

Pro zobrazovani lokaci v google maps se pouzivaji zemepisne souradnice. Ty se z adresy ziskaji pres Google geocoding API. Toto API umoznuje 2500 bezplatnych requestu denne (maximalne 50 za sekundu). Za kazdych dalsich 100 requestu se uctuje 0.50$.

SQL injection:

- pri zadavani do formulare → SQL dotazy se budou tvorit parametrizovanymi queries
- podvrzena data primo v souboru z webu doprava.plzensky-kraj.cz

Prehlceni databaze → automaticke mazani velmi starych dat (zalezi na dohode s krajem)
Cas zpracovani SQL dotazu pro velke casove intervaly → pridani paralelnich dat redukovanych za cely den

8 Testování

- Automatické testy REST api
- Manuální UI testy