FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Zobrazení dat o průjezdu vozidel

KIV/ASWI - Technická dokumentace

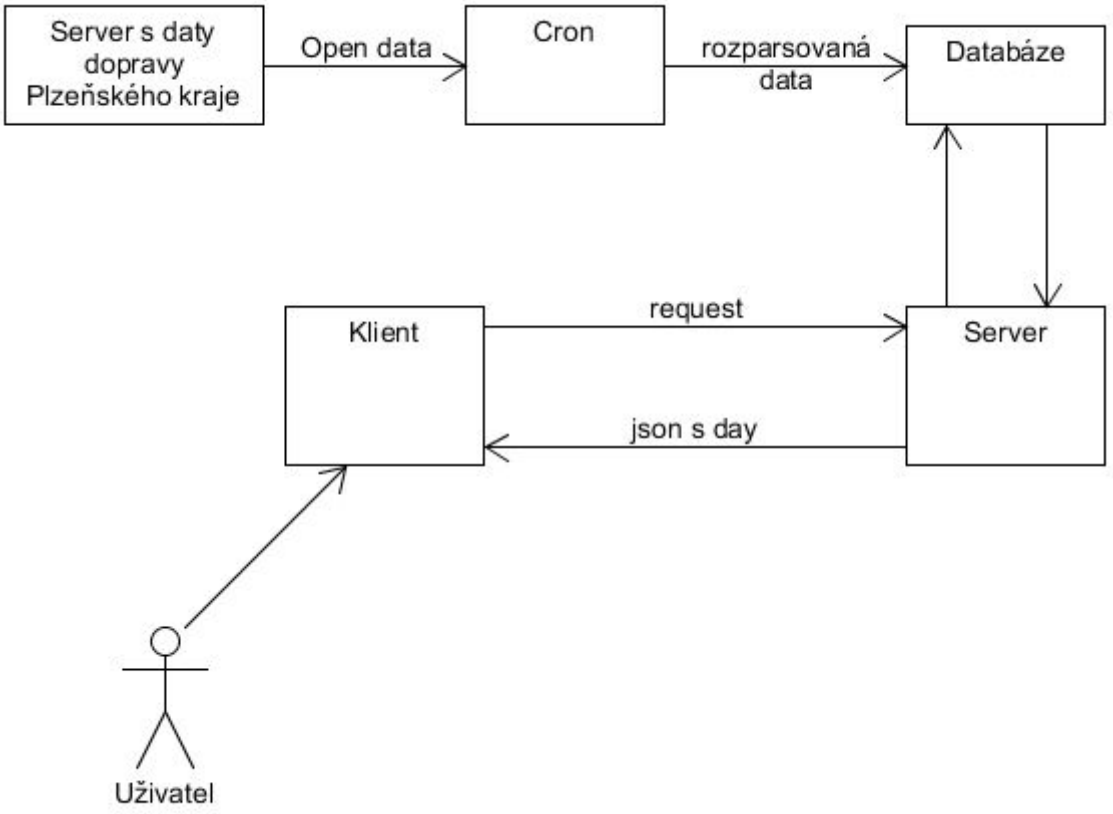
Plzeň, květen 2018

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

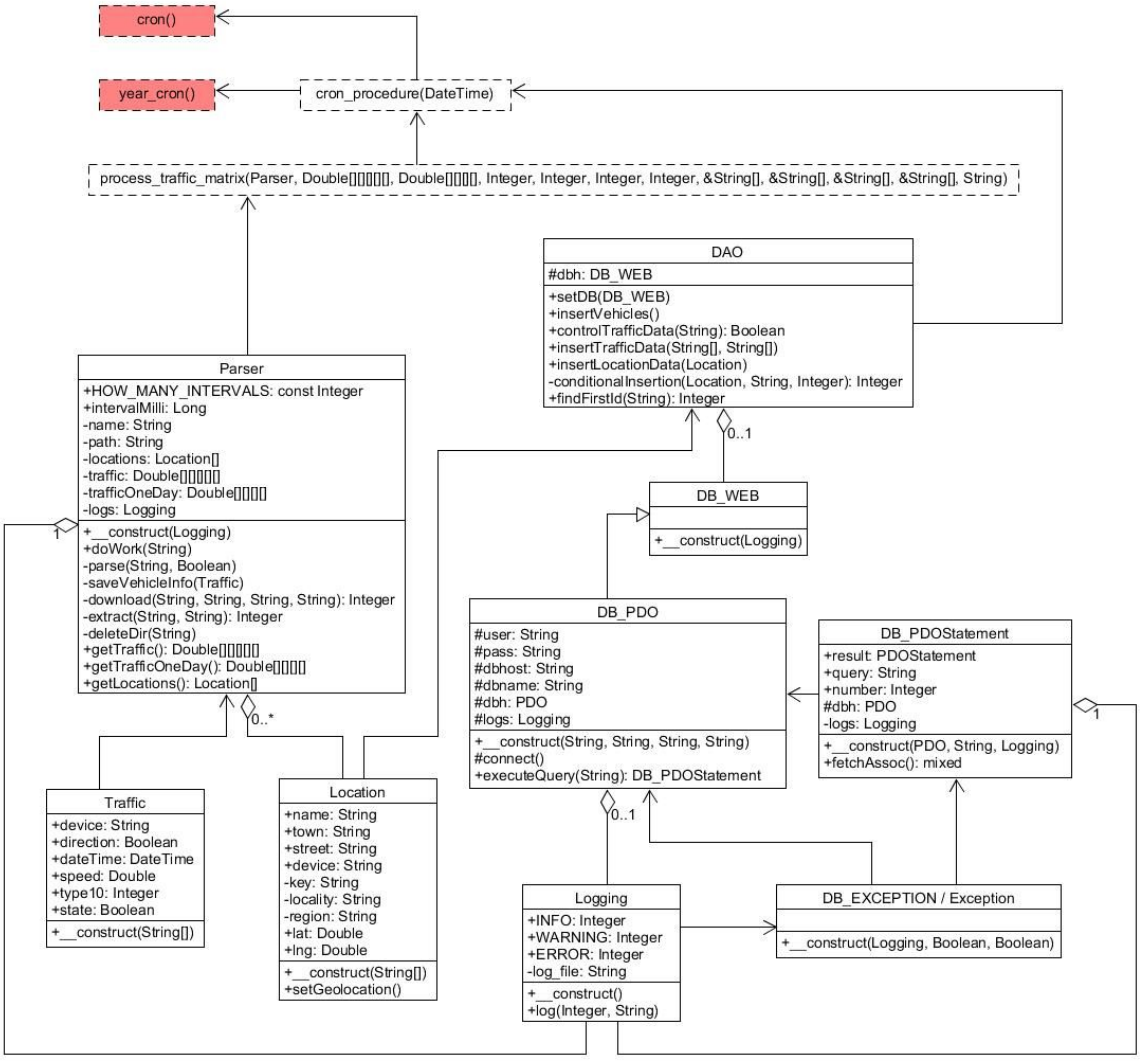
2 Úvod

Blokové schéma architektury aplikace je znázorněno na následujícím obrázku:



3 CRON

Složka `cron` obsahuje skripty pro stažení, rozbalení, zpracování a uložení záznamu o dopravě. Skripty pro cron byly testovány hlavně praktickým využíváním. Ovšem část, která se stará o zpracování dopravních záznamů bylo potřeba otestovat detailně, abychom si byli jistí, že algoritmus funguje tak, jak má. Vše, co je potřeba k otestování (včetně prázdné a naplněné databáze), je ve složce `cron_test`. Jelikož se složka `cron` a `cron_test` moc neliší, nebudu jí tady rozebírat. Nějaké informace jsou v `README` souborech a pak v komentářích v kódu.



1. UML diagram tříd a skriptů (čerchovaně)

3.1 Adresářová struktura a základní popis složek, tříd a skriptů

- **dao**
  Složka obsahující jedinou třídu `DAO`.

  - **dao.php**
    Třída s funkcemi s konkrétními dotazy na databázi.

- **db**

  Složka s třídami pro navázání spojení s databází a vyřízení dotazů.

  - **db-exception.php**
    Třída představující výjimku, která je vyhozena při chybě v komunikaci s databází. Nastavuje se zde kód chyby a zpráva.

  - **db-pdo.php**
    Obsahuje dvě třídy. První třída `DB_PDO` obsahuje funkce pro připojení k databázi a pro vykonání požadavku. Druhá třída `DB_PDOStatement` je využívaná po vykonání určitého dotazu na databázi. Vyhodnocuje úspěšnost vykonání dotazu a v případě problémů vyhazuje výjimku.

  - **db-web.php**
    Obsahuje třídu `DB_WEB`, která dědí od třídy `DB_PDO` a pouze nastavuje atributy potřebné pro připojení k databázi.

- **download**

  Jedná se o pomocnou složku, do které se dočasně ukládá stažený `.zip` soubor s informace o dopravě. Do této složky se extrahují soubory z archivu. Po zpracování dojde k vyčištění složky. Úpravou jedné řádky kódu lze extrahovaná data ve složce zanechat. Pokud by docházelo k zanechávání extrahovaných dat, data by se nacházela ve složce, která by měla stejný název jako stažený archiv, ze kterého se data extrahovala.

- **cron.php**

  Obsahuje dva hlavní spouštěcí skripty (pro den a rok), které zjistí datum(y) a pak volají funkci pro zpracování archivu a uložení dat do databáze.

- **location.php**

  Třída představující lokaci snímače. Ke každé lokaci se, až na atribut „oblast“, ukládají všechny informace z dané řádky souboru s lokacemi snímačů. Navíc se určují geolokace podle Google.

- **parser.php**

  Třída pro stažení a extrahování archivu. Mimo jiné zde dochází k základnímu předzpracování dat. Informace o lokacích se uloží do objektů třídy `Location` do pole `locations[]`. Těchto dat není moc, takže není problém si je všechny uložit. Navíc jsou většinou shodné pro všechny dny. Největší problém nastává u konkrétních záznamů v daném dni. Zde musí dojít k předzpracování tak, aby byly zachovány pouze pro nás důležité údaje. K předzpracování dojde ve funkci `saveVehicleInfo()`, kde je vstupním parametrem objekt třídy `Traffic` obsahující důležité informace z určitého řádku souboru se záznamy. Zredukované informace se ukládají do pěti rozměrného pole `traffic` takovým způsobem, aby došlo k co největší úspoře místa.

  Prvním rozměrem jsou `ID` zařízení. Pro každé zařízení je pak pole s časovými intervaly (celý den se rozdělí na `N` intervalů, u kterých zjišťujeme kolik vozidel jakého typu a jakou rychlostí projeli). Každý časový interval má pole o dvou prvcích představujících směry. Pro každý směr je vytvořeno pole o jedenácti prvcích představujících rozšířené typy vozidel. Na závěr je pro každý typ vozidla vytvořeno pole o dvou hodnotách, které konkrétně znamenají počet vozidel a suma rychlostí. Takže místo ukládání celého řádku souboru dochází pouze k sumaci dvou atributů. Díky pozici v poli pak dokážeme určit, jakým zařízením byly záznamy pořízeny, během jakého časového intervalu, v jakém směru vozidla jela, o jaký typ vozidel se jednalo, kolik jich bylo a jakou rychlost měly dohromady (není problém spočíst jejich průměrnou rychlost).

  Kromě pole `traffic` je zde ještě čtyř-rozměrné pole `trafficOneDay`, které obsahuje to samé jako pole `traffic`, ale je vynechán rozměr pro denní intervaly. Tohle pole slouží pro určení průměrných dat za celý den. Bylo to přidáno kvůli výpočetní náročnosti při vykonávání dotazů.

- **process_traffic_matrix.php**

  Poté, co je vykonáno základní předzpracování dat, musí dojít k tomu závěrečnému. Tedy zrekonstruovat časové intervaly, u kterých jsou nějaké záznamy, do formátu vhodného k uložení do databáze a pouze pro typy vozidel, které v daný interval projeli, vypočíst jejich průměrnou rychlost.

  Jelikož se do databáze nemůžou uložit všechna data na jednou (z důvodu velkého množství) a ukládat je po jednom zabere spoustu času, musí se záznamy před uložení do databáze připravit tak, aby se jednoduše mohly vzít a určitý počet najednou uložit, respektive vložit několik záznamů na jednou do jednoho příkazu `INSERT`. K tomu slouží skript `process_traffic_matrix.php`, který vytváří čtyři pole řetězců, kde každý prvek v poli obsahuje část příkazu `INSERT`, konkrétně v závorkách obsažené hodnoty, které mají být vloženy do databáze.

  V prvním poli `insertDate[]` jsou data pro tabulku datum, v druhém poli `insertRT[]` jsou údaje pro tabulku záznam, ve třetím poli `insertRTT[]` jsou

údaje pro tabulku záznam-čas a ve čtvrtém poli `insertOneDay[]` jsou data pro tabulku záznam-prům-den.

- **traffic.php**

  Každý objekt této třídy představuje jednu řádku souboru se záznamy (představují tedy jeden konkrétní záznam). Objekty si uchovají pouze informace důležité pro další zpracování.

- **logging.php**

  Třída pro logování do souboru `/log/cron.txt`. Zaznamenává úrovně `INFO`, `WARNING`, `ERROR`. Formát souboru: čas zaznamenání zprávy `<ENTER>` úroveň `<ENTER>` zpráva.

3.2 Detaily jednotlivých tříd

```
class DB_WEB extends DB_PDO {

      protected $user = "root"; * Přihlašovací jméno. *
      protected $pass = ""; * Heslo uživatele. *
      protected $dbhost = "localhost"; * Adresa serveru. *
      protected $dbname = "prujezd_vozidel"; * Název databáze. *
      protected $logs = NULL; * Objekt pro logování. *

      * Konstruktor třídy DB_WEB. *
      public function __construct($logs) {}

}

class DB_Exception extends Exception {

      *
      Konstruktor třídy DB_EXCEPTION nastavuje kód výjimky, zprávu výjimky a zapisuje
      informace do logovacího souboru.
      *
      public function __construct($logs = NULL, $message = false, $code =
      false) {}

}

class DAO {

      protected $dbh; * Reference na objekt třídy DB_WEB. *

      * Nastavuje referenci na vytvořený objekt DB_WEB. *
      public function setDB($dbh) {}
```

```
    * Naplní tabulku vozidla, pokud v tabulce nejsou žádná data. *
    public function insertVehicles() {}

    *
    Kontrola počtu záznamů v tabulce vozidla pro daný den. Vrací TRUE v případě, že v
    tabulce pro dané datum nejsou žádné záznamy.
    *
    public function controlTrafficData($dateStr) {}

    *
    V první iteraci se ukládají data insertRTT do tabulky zaznam_cas, v druhé se ukládají
    insertRT do tabulky zaznam. Najednou se uloží maximálně 500 záznamů do dané
    tabulky.
    *
    public function insertTrafficData($insertRTT, $insertRT) {}

    *
    Pokud daná lokace není v databázi, dojde k jejímu uložení. Testuje se existence města,
    ulice a na závěr zařízení.
    *
    public function insertLocationData($location) {}
    *
    Funkce pro ukládání ulic a měst. Jedná se o podmíněné uložení do databáze (pouze
    pokud dané informace ještě nejsou v databázi). Vrací ID záznamu – jak v případě, že
    záznam v tabulce existoval, tak v případě, že se do tabulky teprve uložil.
    *
    private function conditionalInsertion($location, $table, $townId) {}

    * Zjistí hodnotu ID dalšího záznamu pro danou tabulku a vrátí ji. *
    public function findFirstId($table) {}

}

class DB_PDO {

    protected $user = "root"; * Přihlašovací jméno. *
    protected $pass = ""; * Heslo uživatele. *
    protected $dbhost = "localhost"; * Adresa serveru. *
    protected $dbname = "prujezd_vozidel"; * Název databáze. *
    protected $dbh; * Reference na objekt přímo komunikující s databází – PDO. *
    protected $logs; * Reference na objekt pro logování. Nastavuje se až v DB_WEB. *

    * Nastavení instančních atributů. *
    public function __construct($user, $pass, $dbhost, $dbname) {}

    * Připojení k databázi. Pokud se k databázi nejde připojit, vyhodí výjimku. *
    protected function connect() {}
```

```
        *
        Vykonání dotazu nad databází. Pokud není vytvořené připojení k databázi, zavolá se
        funkce connect(). Pokud se dotaz neprovede, dojde k vyhození výjimky. Vrací referenci
        na DB_PDOStatement obsahující informace o zpracování dotazu.
        *
        public function executeQuery($query) {}

}

class DB_PDOStatement {

        public $result; * Výsledek vykonání dotazu. *
        public $query; * Dotaz, který měl být vykonán. *
        public $number; * Počet výsledných záznamů. *
        protected $dbh; * Reference na objekt PDO. *
        private $logs; * Reference na objekt pro logování. *

        *
        Kromě nastavení instančních atributů navíc, pokud neexistuje spojení s databází,
        vyhodí výjimku, které předá referenci na objekt pro logování.
        *
        public function __construct($dbh, $query, $logs) {}

        *
        Pokud dotaz nebyl vykonán, vyhodí výjimku. Jinak vrátí výsledek vykonání dotazu.
        *
        public function fetchAssoc() {}

}

class Parser {

        public $HOW_MANY_INTERVALS = 96; * Na kolik intervalů se má den rozložit. *
        public $intervalMilli; * Délka v milisekundách jednoho intervalu. *
        private $name; * Začátek názvu jakéhokoli archivu. *
        private $path; * URL na které se nachází archivy. *
        private $locations; * Pole lokací. *
        private $traffic; * Pěti rozměrné pole s informacemi o dopravních záznamech. *
        private $trafficOneDay; * Čtyř-rozměrné pole s průměry za celý den. *
        private $logs; * Objekt pro logování do souboru cron.txt ve složce log. *

        * Pouze nastavuje instanční atributy. *
        public function __construct($logs) {}

        *
        Volá funkce pro stažení a extrahování archivu, pro jeho zpracování a na závěr volá
        funkci pro odstranění složky daného archivu.
        *
        public function doWork($date) {}
```

```
    * Má na starosti zpracování obou souborů (soubor s lokacemi a soubor se záznamy). *
    private function parse($fileName, $traffic) {}

    * Naplnění pole traffic. Viz algoritmus výše. *
    private function saveVehicleInfo($t) {}

    *
    Tahle funkce vytvoří ve složce download složku pro archiv a provede jeho stažení. Funkce vrací hodnoty <-3; 0>. Jediná nechybová je 0 a znamená, že celá procedura stažení proběhla v pořádku.
    *
    private function download($date, $zipUrl, $dir, $downloaded) {}

    * Extrahuje archiv. Vrací 0, pokud vše proběhlo v pořádku, jinak -1. *
    private function extract($dir, $downloaded) {}

    * Odstraní jakoukoli složku včetně celého obsahu. *
    private function deleteDir($path) {}

    * Getter pro atribut traffic. *
    public function getTraffic() {}

    * Getter pro atribut trafficOneDay. *
    public function getTrafficOneDay() {}

    * Getter pro atribut locations. *
    public function getLocations() {}

}

class Location {

    public $name; * Popis umístění. *
    public $town; * Název města. *
    public $street; * Název ulice. *
    public $device; * ID detektoru. *

    private $key; * Klíč pro získání JSON s geolokacemi. *
    private $locality; * Pro který kraj se mají geolokace vyhledávat. *
    private $region; * Pro kterou zemi se mají geolokace vyhledávat. *

    public $lat; * Zeměpisná šířka. *
    public $lng; * Zeměpisná délka. *

    * Konstruktor třídy Location, do kterého vchází rozdělená řádka ze souboru lokací. *
    public function __construct($data) {}

    * Nastaví zeměpisnou šířku a délku dané ulice. *
    public function setGeolocation() {}

}
```

```
class Traffic {

      public $device; * ID detektoru. *
      public $direction; * Směr snímání. *
      public $dateTime; * Datum a čas pořízeného záznamu. *
      public $speed; * Rychlost v km/h. *
      public $type10; * Plné označení typu vozidla. *
      public $state; * Stav detektoru. *

      * Konstruktor třídy Traffic, do kterého vchází rozdělená řádka ze souboru záznamů. *
      public function __construct($data) {}

}

class Logging {

      const INFO = 0; * Informativní zprávy. *
      const WARNING = 1; * Varování. *
      const ERROR = 2; * Kritické chyby. *
      private $log_file; * Název a cesta k souboru pro logování. *

      * Konstruktor třídy Logging, který pouze nastaví název a cestu k souboru. *
      public function __construct() {}

      * Na konec souboru přidá záznam s danou úrovní a zprávou. *
      public function log($type, $message) {}

}
```

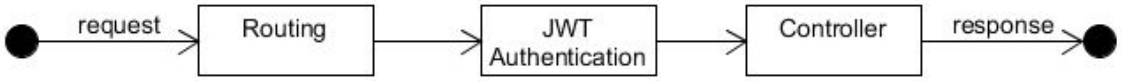
4 Backend

Serverová část aplikace má za úkol skrze REST API poskytovat poskytovat data k vizualizaci klientské části. Kód serverové části je umístěn ve složce `backend`. K realizaci byl zvolen framework Lumen verze 5.2, která je kompatibilní s PHP verze 5.5.9 a vyšší.

Z adresářové struktury jsou nejdůležitější: soubor `.env`, který obshauje konfiguraci (přístupové údaje k databázi, parametry autentizace), složka `public`, která obsahuje index.php pro přístup k serveru a složka `app`, která obsahuje obslužný kód (kontroléry, modelové třídy).

4.1 Zpracování požadavku

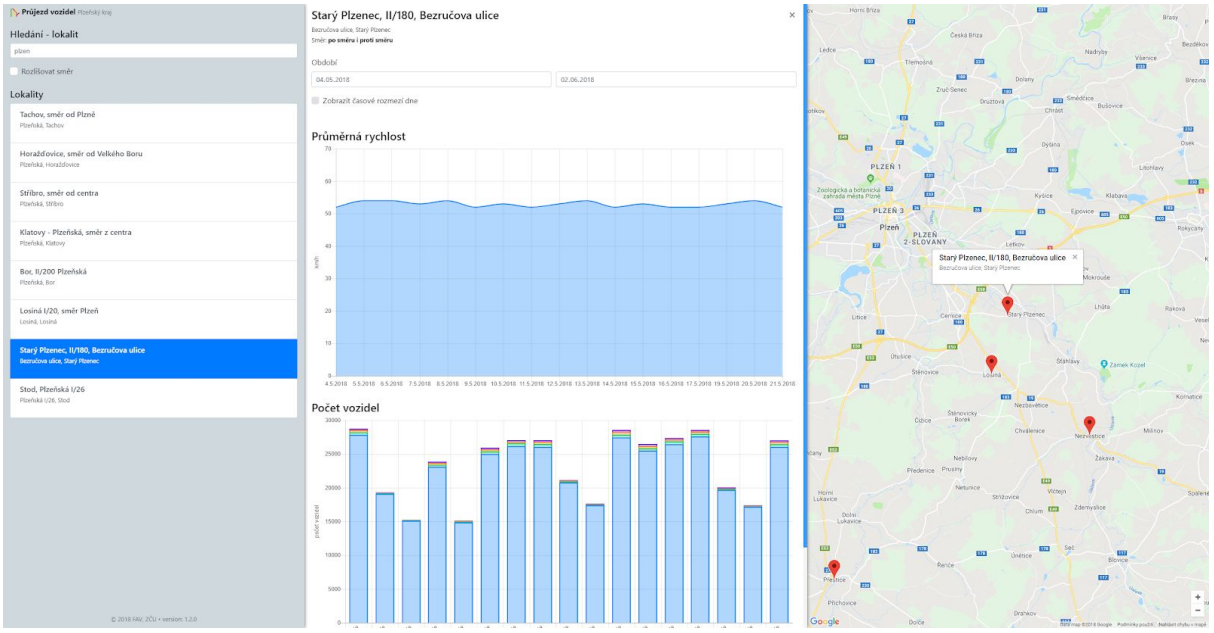
Schéma zpracování požadavku je znázorněno na obrázku níže. Každý request na validní url prochází JWT autentizací (kterou je možné vypnout). Po úspěšné autentizaci je zpracován v kontrolleru.



Nastavení api endpointů (url) je provedeno v souboru `app/Http/routes.php`.

5 Frontend

Pro rozvržení layotu byl použit Bootstrap 4.1 a preprocesor SASS.
Layout se skládá ze tří sloupců vyhledávání, info a mapa.
Pravý sloupec vyhledávání nabízí dynamické vyhledávání detektorů podle adresy, checkbox “Rozlišovat směr” umožňuje zobrazit oba směry.
Prostředním sloupcem je info, kde se zobrazují informace a grafy k aktuálně vybraném detektoru.
Tento sloupec lze jako jediný zavřít, aby uvolnil místo mapě.
Poslední a levý sloupec mapa zobrazuje umístění detektoru v plzeňském kraji.



Také obsahuje loading screen, který se schová po načtení stránky. Dalším prvkem je modalní okno, které se zobrazuje jen při chybách jako je vypršení tokenu.

Data-binding zajišťuje AngularJS ve verzi 1.6.10, každý sloupec má svůj vlastní kontrolér a plus hlavní, který se o celou stránku.

Čtyři kontroléry:

- searchController - stará o vyhledání a vypsaní výsledků
- infoController - rendrováni grafu do canvas a validaci časového rosahu
- mapController - zajišťuje vykreslení v detektorů na mapě, zazoomování na vybraný detektor a odzoomovaní do výchozí pozice
- mainController - obstarává zobrazení modalního okna, skrývání loading screen a reaguje na změny v url

Knihovny:

- Google Maps API - zobrazuje mapu a zakresluje do ní
- Char.js - vykresluje grafy do canvas
- Moment.js - zjednodušuje práci s časem

V konfiguraci lze nastavit:

- APP_VERSION - obsahuje verzi, která se ukáže uživateli
- API_URL - základní url REST API
- API_TOKEN - token je vždy nově vygenerován před odesláním stránky
- DEFAULT_POSITION - obsahuje výchozí hodnotou jsou souřadnice plzeňského kraje
- DEFAULT_ZOOM - obsahuje zazoomování po načtení, výchozí hodnota je 10
- DEFAULT_ZOOM_MAX - obsahuje limit maximalního odzoomování, výchozí hodnota je 7