Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Zobrazení dat o průjezdu vozidel z kamerového a radarového systému Plzeňského kraje

Plzeň 2018 Václav Jirák

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 27. června 2018

Václav Jirák

Poděkování

Rád bych tímto poděkoval vedoucímu bakalářské práce panu Ing. Tomáši Potužákovi, Ph.D. za ochotu, cenné rady a čas, který mi věnoval při vypracování bakalářské práce.

Abstract

Result of this thesis is an application that allows visualization of freely available road data from camera and radar system of Pilsen region. The thesis deals with data visualization and tools for displaying arbitrary data. Design and implementation of the resulting application are then described. The main focus of the application is on interactive and user-friendly visualization.

Abstrakt

Výsledkem této práce je aplikace umožňující vizualizaci volně dostupných silničních dat z kamerového a radarového systému Plzeňského kraje. Je probrána problematika vizualizace dat a nástroje umožňující zobrazení vlastních dat. Následuje popis návrhu a implementace výsledné aplikace, která se zaměřuje především na interaktivní a uživatelsky přívětivou vizualizaci.

Obsah

1 Úvod

Cílem této práce je vytvoření otestované aplikace pro online vizualizaci volně dostupných dat o projíždějících vozidlech na silnicích. Tyto data jsou pořizována z kamerového a radarového systému Plzeňského kraje. Aplikace by měla být schopna data zpracovat a následně vizualizovat ve vhodné podobě.

V práci budou v první kapitole probrány možnosti vizualizace obecných dat a následně se kapitola zaměří na vizualizaci dat ze silniční dopravy. Dále se bude pojednávat o vizualizačních nástrojích umožňující tvorbu grafů.

Druhá kapitola se bude nejdříve týkat způsobů zobrazení dat na mapě. Poté budou vysvětleny termíny důležité pro online mapové portály a nakonec budou probrány vybrané mapové portály a jejich možnosti pro zobrazení vlastních dat.

Další kapitola se bude zabývat návrhem samotné aplikace. Bude zde popsáno pojetí aplikace, a jaké technologie byly vybrány, jelikož jak pojetí aplikace, tak použití konkrétních technologií nebylo součástí zadání práce. Dále budou podrobně rozebrány aplikací zpracovávaná data a následný způsob jejich stahování, zpracování a uložení. Poté budou zvoleny vhodné způsoby vizualizace zpracovávaných dat a bude navrhnuto grafické uživatelské rozhraní. Na konci této kapitoly bude navrhnuta architektura aplikace.

Předposlední kapitola se bude týkat implementace. Budou vysvětleny implementační detaily a popsány jednotlivé třídy a moduly. Nakonec bude popsána komunikace mezi klientem a serverem.

Poslední kapitola se bude týkat naměřených výsledků a testování výsledné aplikace. Budou zde popsány výsledky, podle kterých se v určité části realizace vybíral způsob řešení. Následně se bude kapitola týkat způsobů, jakými byla aplikace otestována.

2 Vizualizace dat

2.1 Vizualizace obecných dat

Mezi základní cíle vizualizace dat patří převod dat do grafického formátu, se kterým lze pracovat mnohem efektivněji, než kdybychom pracovali s původními daty. Existuje několik možností, jak mohou být data vizualizována, z nichž se na základě typu dat, která chceme vizualizovat, vybírá ta nejvhodnější, nebo několik nejvhodnějších.

2.1.1 Typy dat

Jedním z rozdělení typů dat je rozdělení na data kvalitativní neboli kategorická a kvantitativní neboli číselná. Kategorická data jsou data popisující povahu údajů, jako například identifikační číslo, jméno, či odpověď v průzkumu, jako je ano či ne. Kvantitativní data jsou data numerická. Patří mezi ně data typu rychlost v kilometrech za hodinu, čas v sekundách nebo hmotnost v kilogramech. Kvantitativní data se dále dělí na data spojitá a data diskrétní, kde spojitá data tvoří kontinuum nekonečného počtu kroků, jako například čas a velikost. Naproti tomu diskrétní data obsahují konečné hodnoty, které lze spočítat, jako například počet studentů na fakultách. Mohlo by se zdát, že všechna numerická data jsou kvantitativní, ale není tomu tak, jelikož například telefonní číslo nebo identifikační číslo jsou numerická data, ale řadí se mezi data kategorická [1].

2.1.2 Gestaltismus

Důležitým pojmem v oblasti vizualizace dat je tzv. gestaltismus neboli tvarová psychologie. Pojem gestaltismus vychází z německého výrazu Gestalt, což lze do češtiny přeložit jako podoba, tvar nebo celek. Z gestaltismu vyplývá, že elementy ve vizualizaci mohou být vnímány jako celek. Jako jeden celek mohou být vnímány například prvky oddělené úzkou mezerou nebo prvky se stejnou barvou. Prvky lze pomocí barev oddělit i podle důležitosti, kde ale nastává problém, jakou barvu nastavit jako nedůležitou a jakou barvu nastavit jako důležitou, protože zde hraje roli vliv kulturních zvyklostí. Všude nemusí nutně znamenat, že například zelená je “méně důležitá” než červená.

Tento problém se dá vyřešit například tak, že se vybere jedna barva a důležitost se zdůrazňuje sytostí barvy nebo se k vizualizaci přidá legenda, kde je zobrazena „škála důležitosti“. Gestaltismus se skládá z následujících šesti principů [2]:

- *Vzdálenost* – objekty, které jsou si bližší, jsou vnímány jako celek
- *Podobnost* – objekty, které jsou si podobné, jsou vnímány jako celek
- *Spojitost* – lidé v obrazcích hledají čáry s nepřerušeným pokračováním
- *Uzavření* – objekty jsou vnímány jako uzavřené, i když ve skutečnosti nejsou
- *Plocha* – překrývají-li se dva objekty, je jeden objekt vnímán, jako by byl na popředí druhého
- *Symetrie* – objekty se spojují podle symetrie

2.1.3 Možnosti vizualizace

Mezi základní možnosti, jak data vizualizovat, patří zobrazení dat pomocí tabulky, zobrazení dat pomocí grafu nebo, pokud jsou data spjatá s geografickou polohou, vizualizace dat na mapě. V následujících podkapitolách bude probrána problematika vizualizace tabulkou a grafem. Vizualizace na mapě se týká *Kapitola 3.1*.

2.1.4 Vizualizace tabulkou

Pokud transformujeme vstupní data do matice, hovoříme o vizualizaci tabulkou. Tabulky obsahují data organizovaná do sloupců a řádků, kde například jeden řádek může představovat jednu osobu a jednotlivé sloupce mohou představovat jméno, pohlaví, věk a další informace o dané osobě. Tabulky jsou vhodné v případech, kdy se porovnávají individuální hodnoty a je-li znalost hodnot klíčová.

Tabulka 2.1: Ukázka přidání dodatečného sloupce „Změna v %“ obsahující procentuální změnu návštěvnosti

Město	Počet návštěv v květnu 2012	Počet návštěv v červnu 2012	Změna v %
Londýn	31 733	81 077	+155,50%
New York	9 451	8 591	-9,10%
Manchester	6 395	7 797	+21,92%
Nové Dillí	3 879	5 435	+40,11%

Do tabulky je možno také přidat sloupec obsahující výsledek nadefinované dodatečné funkce, čímž lze poukázat na nějaký trend, viz *Tabulka 2.1*, kde je přidán sloupec „Změna

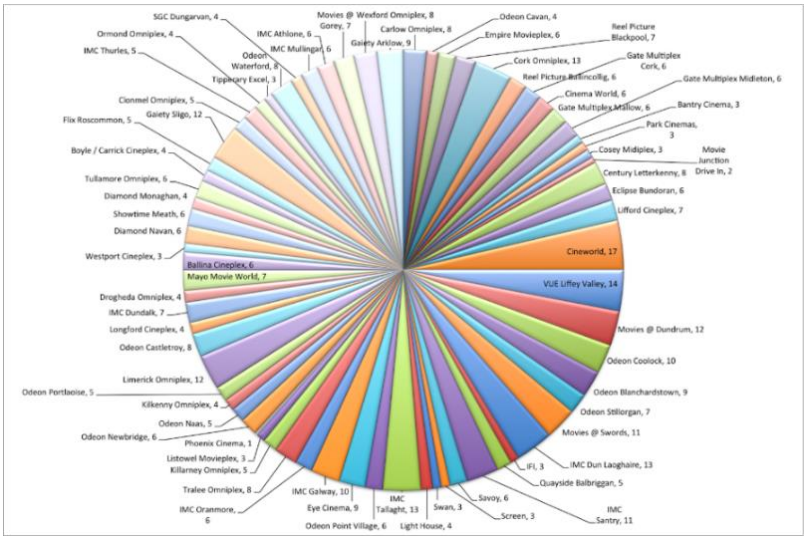
v %“ znázorňující procentuální změnu návštěvnosti jednotlivých měst mezi květnem a červnem roku 2012 [2].

2.1.5 Vizualizace grafem

Graf zobrazuje data pomocí vizuálních objektů, jako například sloupců ve sloupcovém grafu, bodů v bodovém grafu nebo výsečemi kruhu ve výsečovém grafu. Použití grafu je vhodné zejména, pokud se chce poukázat na vztah mezi daty, kdy je při správě zvoleném grafu na první pohled vidět nějaký vzor nebo trend. Graf se typicky skládá z několika částí [2]:

- Obsah – sloupce, čáry, body, …
- Osy, jejich popisky a hodnoty
- Titulek
- Legenda – například u výsečového grafu popis, jaká barva patří jaké kategorii

Existuje mnoho typů běžně používaných grafů. Mezi základní patří například výsečový graf, sloupcový graf, spojnicový graf a bodový graf.

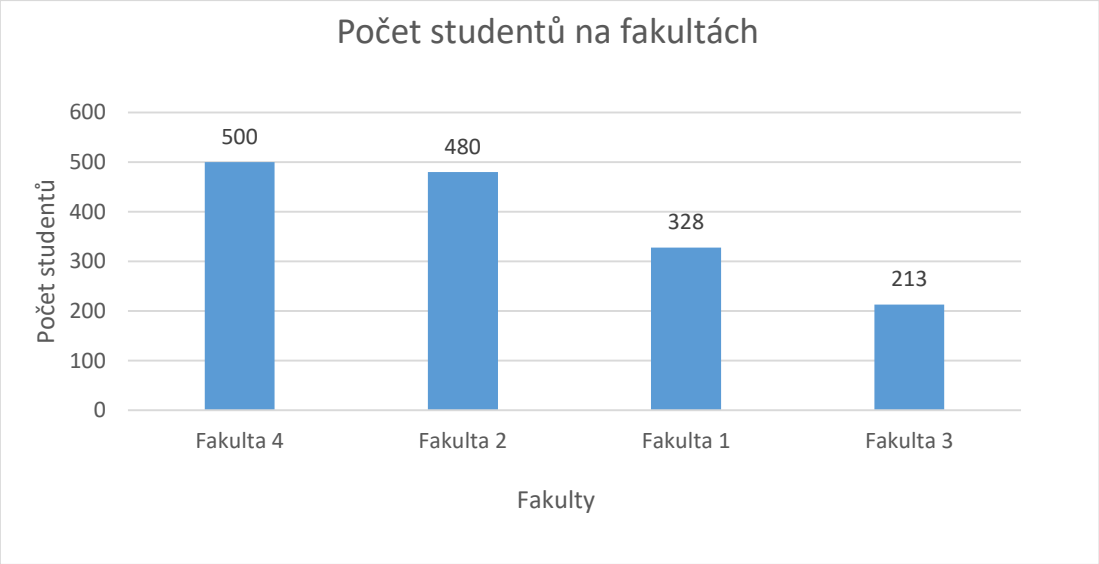


Obrázek 2.1: Obrázek poukazující na nevhodnost užití výsečového grafu při mnoha položkách
Zdroj: `https://briancort.com/wp/wp-content/uploads/2017/03/badPie.png`

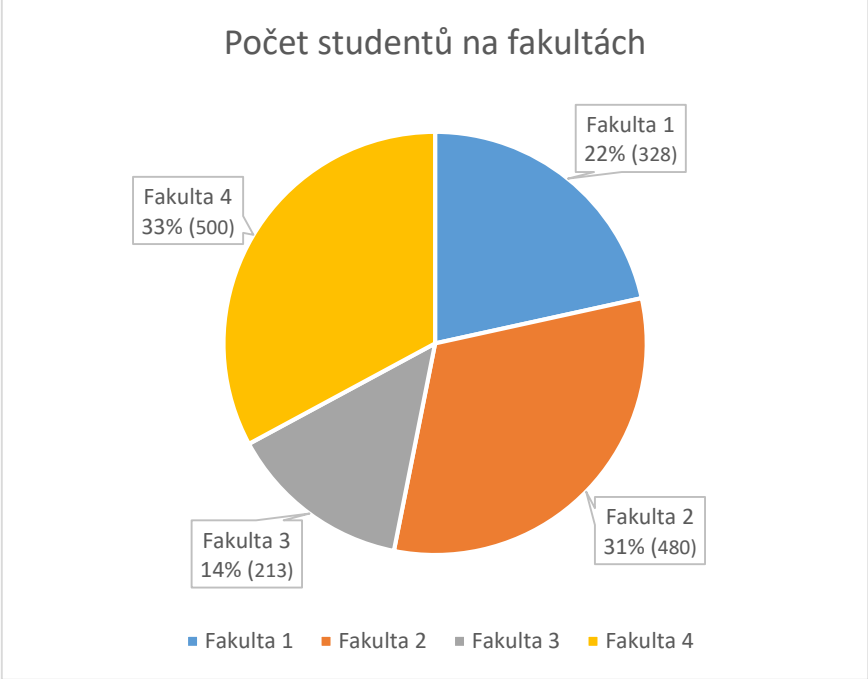
Pro určitý typ dat se hodí pouze určité typy grafů. Často lze data vizualizovat více typy grafů, ale pouze pomocí některých z nich docílíme srozumitelné vizualizace. Zobrazuje-li se například proporcionální zastoupení, lze použít jak výsečový graf, tak sloupcový graf a volba mezi těmito dvěma grafy záleží na povaze dat. Výsečový graf se doporučuje používat, je-li počet položek roven nebo menší než čtyři, v opačných případech je

vhodnější přiklonit se ke grafu sloupcovému, jinak se může graf stát nepřehledným, viz *Obrázek 2.1* [2].

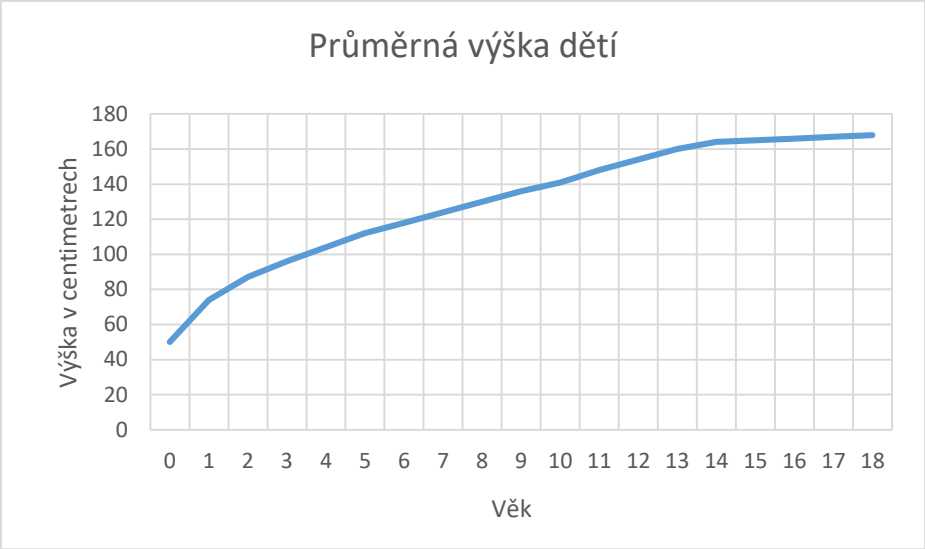
Dále je také nutno brát v potaz, zda jsou data spojitá nebo diskrétní. Pro diskrétní data, jako počet studentů na fakultách, není vhodné použít například spojnicový graf, ale graf sloupcový, případně výsečový, viz *Obrázek 2.2* a *Obrázek 2.3*. Naopak pro hodnoty, jako je průměrná výška dětí, je použití tohoto grafu vhodné, viz *Obrázek 2.4*.



Obrázek 2.2: Sloupcový graf zobrazující počet studentů na jednotlivých fakultách



Obrázek 2.3: Výsečový graf zobrazující počet studentů na jednotlivých fakultách



Obrázek 2.4: Spojnicový graf zobrazující průměrnou výšku dětí

2.1.6 Interaktivní vizualizace

U rozsáhlých dat je k efektivní vizualizaci někdy vhodné přidat interaktivitu. Nevýhodou přidáním interaktivity je, že ztrácíme možnost jednoduchého vytisknutí kompletní vizualizace na papír. Mezi nejdůležitější aspekty pro kvalitní interaktivní vizualizaci patří ovládací prvky, kterými si zobrazujeme dodatečné informace, či manipulujeme s vizualizací a doba odezvy, za jak dlouho se interakce projeví [2].

Interaktivitu lze přidat několika způsoby. Například přidáním posuvníku, na kterém se navolí, z jakého dne se budou data zobrazovat nebo přidáním možnosti, kdy se po kliknutí na značku, vykreslené na mapě, zobrazí dodatečné informace.

2.2 Vizualizace dat ze silniční dopravy

V dnešní době, kdy v České republice připadá téměř jedno vozidlo na dva obyvatele [6], je žádoucí, abychom si mohli například cestu naplánovat za účelem pohodlné dopravy. Pro efektivní naplánování cesty není jediným faktorem jaká cesta je nejkratší nebo nejrychlejší, ale také například stav daných komunikací a hustota provozu. Znalostí těchto faktorů se můžeme vyhnout zácpám, případně i nehodám. A právě vizualizací faktorů, jako například hustota provozu, se zabývá vizualizace silničních dat. Vizualizace silničních dat nemusí být nutně určena pouze pro plánování cest. Lze jí využít například k analýze, zda by nebylo vhodné přidat další jízdní pruh, k optimalizaci světelných

křižovatek, či zjištění, kde by mohlo být nejvíce znečištěné ovzduší způsobené motorovými vozidly.

2.2.1 Silniční data

Silniční data mohou být naměřena několika způsoby, jako například vygenerováním záznamu senzory, rozmístěných podél pozemních komunikací nebo vygenerováním záznamu ze senzorů, zabudovaných přímo ve vozidlech. Módy, ve kterých mohou senzory pracovat, lze rozdělit následovně [7]:

- *Založený na poloze* – Záznam je vygenerován, pokud vozidlo vjede do rozsahu senzoru
- *Založený na aktivitě* – Senzor vygeneruje záznam, pokud nastane určitá aktivita, například pokud vozidlo překročí limitní rychlost
- *Založený na zařízení* – Senzor je umístěn ve vozidle a aktivně generuje záznamy

Kapitola se bude dále zabývat pouze senzory v módech založených na poloze a na zařízení. Jeden záznam může obsahovat několik informací. Například otevřená dopravní data Plzeňského kraje [8] obsahují mimo jiné informace typu – kdy byl záznam pořízen, rychlost daného vozidla, typ vozidla a směr ze kterého k senzoru vozidlo přijelo. U těchto záznamů je geografická poloha z jednoho senzoru pevně daný bod, jelikož senzory pracují v módu založeném na poloze. U dat získaných senzory v módu založených na zařízení, je většinou součástí záznamu i geografická poloha [7].

2.2.2 Předzpracování dat

Často je nutné před samotnou vizualizací data nějakým způsobem předzpracovat. Předzpracování je někdy nutné z výkonnostních důvodů, které jsou relevantní hlavně u interaktivní vizualizace, kde se provádí nad danými daty nějaké filtrování a výpočty v reálném čase a je žádoucí, co možná nejmenší odezva systému. Dále z analytických důvodů, kdy je například vhodné opravit, či odstranit chybně zaznamenaná nebo duplicitní data. Předzpracování silničních dat se zpravidla skládá z čištění, přiřazení, agregace a organizace dat [7]. Jednotlivé kroky jsou popsány v následujících odstavcích.

Pod pojmem čištění dat rozumíme odstranění chyb nebo zpracování odlehlých a konfliktních hodnot. Čištění dat se obvykle skládá ze tří fází. V první fázi se hledají

chyby, mezi které se řadí například neunikátnost, chybný pravopis nebo protichůdné hodnoty. Další fáze, která může být provedena manuálně i automaticky, se stará o navrhnutí, jak by dané chyby šly opravit. Poslední fází je samotné provedení navržené opravy [7].

Další krok, přiřazení dat, se používá u dat zaznamenaných ze senzorů pracujících v módu založeném na zařízení. Senzor v tomto módu generuje data, která obsahují geografickou polohu ve formě diskrétních bodů, které ale nemusí ležet kvůli chybám na digitální mapě přesně na silnici. A právě odstraněním těchto chyb neboli zarovnáním bodů na silnici, se zabývá tento krok [7].

Třetím krokem je agregace dat. Agregací dat lze zredukovat obsáhlost dat, někdy samozřejmě za cenu ztráty některých informací. Základní agregované operace lze rozdělit na prostorové, časové a směrové. Prostorové agregace se provádí především výpočtem hustoty datových bodů v daných částech mapy, časové agregace rozdělují data do časových intervalů a směrové agregace agregují směr [7].

Posledním krokem předzpracování dat je organizace dat. Předzpracovaná data musejí být vhodným způsobem organizována. K tomu lze použít databázi, ve které se dá využít k urychlení vyhledávání a řazení dat například indexů [7].

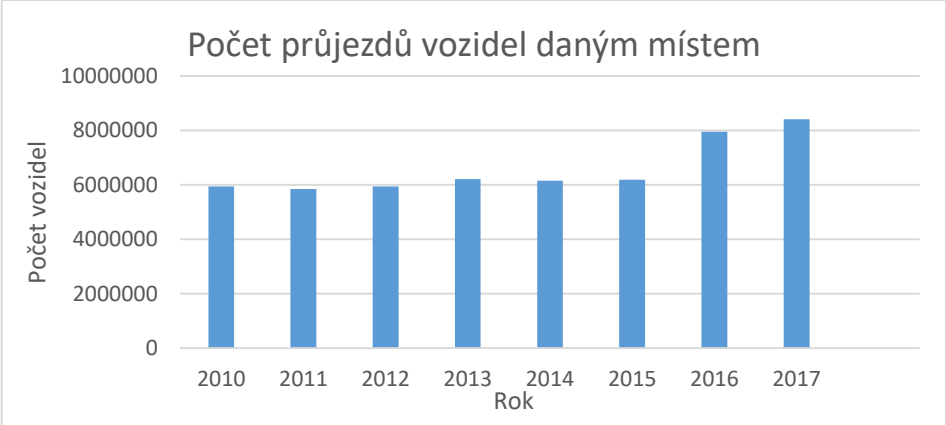
2.2.3 Vizualizace silničních dat

Úkoly vizualizace silničních dat se dají rozdělit do několika kategorií. Jednou z nich je vizuální monitoring dopravních situací, čímž lze odhalit jinak skryté události, jako například dopravní zácpy. Dalším důležitým úkolem je plánování a doporučení cest. Mezi další úkoly se řadí objevování vzorů v dopravě a predikce dopravních událostí [7].

Hlavním faktorem při volbě typu vizualizace je, v jakém módu senzoru byla silniční data pořízena a jaká složka dat se bude zobrazovat. Silniční data se typicky skládají z několika složek, kde mezi nejdůležitější patří čas a poloha [7]. Dalšími složky mohou být například rychlost a typ vozidla. V této kapitole se dále budou popisovat vizualizační techniky pro zobrazení vyjmenovaných složek a jejich kombinací.

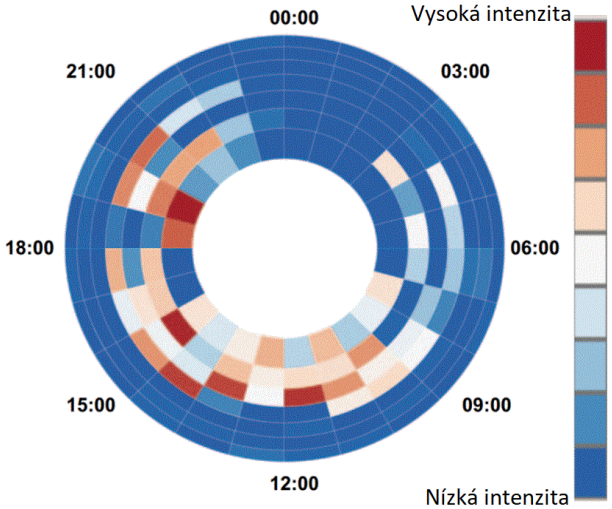
Vizualizací kombinace časové složky a další vybrané složky se typicky snažíme zobrazit nějaký trend, periodicitu nebo abnormality závislé na času. Čas lze obecně rozdělit na lineární, periodický a větvený, který se ale ve vizualizaci silničních dat nepoužívá [7].

Lineární čas je lineární pole od počátečního času ke konečnému času a dokáže vyjádřit, jak se silniční data mění s časem [7]. Obsahují-li data časovou a polohovou složku, lze například vizualizovat, kolik vozidel projelo za uplynulé roky daným místem, na což je vhodné použít sloupcový graf, viz *Obrázek 2.5*. Pokud by data obsahovaly navíc rychlostní složku, šly by sloupce v *Obrázku 2.5* nahradit průměrnou rychlostí za rok, čímž by se ukázalo, jak se liší průměrná rychlost mezi roky.



Obrázek 2.5: Vizualizace počtu průjezdů vozidel daným místem za jednotlivé roky sloupcovým grafem

Periodickým časem se rozumí nějaký časový interval, který se pravidelně opakuje, jako jsou roční období. Periodický čas společně s další složkou, jako složkou polohovou, se dá vizualizovat například takzvaným radial layoutem, viz *Obrázek 2.6,* kde lze zpozorovat, jak se mění hustota dopravy na jednom místě během hodin jednotlivých dnů.



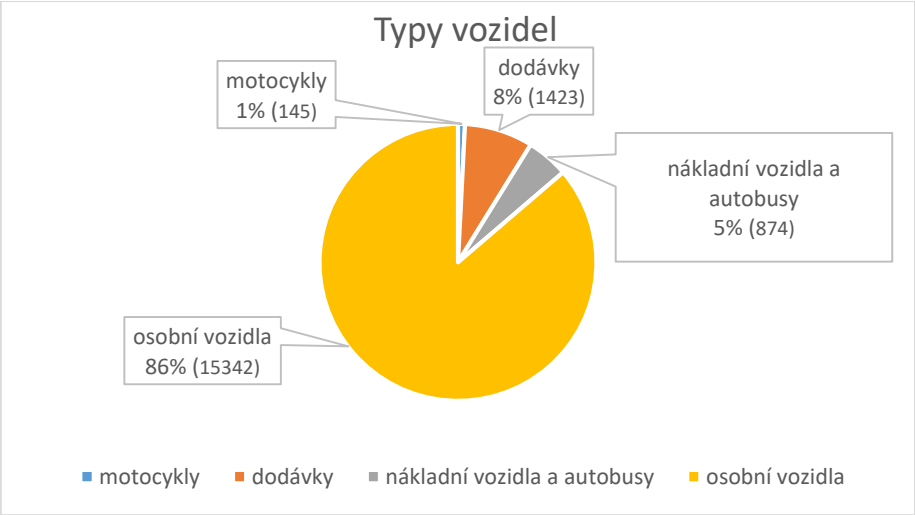
Obrázek 2.6: Vizualizace hustoty dopravy radial layoutem, kde jednotlivé kružnice představují dny v týdnu a části kružnice představují jednu hodinu. Barva části kruhu zobrazuje hustotu dopravy.
Zdroj: `https://ieeexplore.ieee.org/document/7120975/`

Pro vizualizaci pouze polohové složky nebo kombinace polohové složky se složkou další, například rychlostní, je efektivní použít vizualizaci na mapě, na kterou se zaměřuje *Kapitola 3.1*. Vizualizaci polohové složky silničních dat na mapě lze rozdělit na bodovou, čárovou a regionální [7].

Bodová vizualizace zobrazuje záznamy jako body na mapě. Bod může mít různý význam. Jsou-li data pořízená senzory v módu založeném na zařízení, pak bod představuje přímo neagregovaná naměřená data [7]. Jsou-li data pořízená senzory v módu založeném na poloze, pak má bod většinou přidanou doplňkovou grafickou vlastnost, jako barvu, či velikost, viz *Kapitola 3.1*, která může znázorňovat například počet vozidel, či průměrnou rychlost vozidel zaznamenaných daným senzorem.

Čárová vizualizace zobrazuje záznamy jako křivky představující části pozemní komunikace. Křivky jsou opět doplněny grafickou vlastností představující například, jak je daná pozemní komunikace frekventovaná. U dat pořízených senzory v módu založeném na poloze se jedná o totožnou vizualizaci, jako u bodové vizualizace s tím rozdílem, že je bod transformován na křivku, tedy bodová poloha záznamu je změněna na křivku. U dat ze senzorů v módu založeném na zařízení navíc probíhá polohová agregace, tedy křivka agreguje záznamy, které se s ní polohově překrývají.

Regionální vizualizace sumarizuje záznamy z určitých regionů do polygonů představujících například kraje České republiky. Zde se může agregovat polohová složka i u senzorů v módu založeném na poloze, jelikož v jednom regionu může ležet více senzorů.



Obrázek 2.7: Vizualizace typů vozidel výsečovým grafem

Poslední složkou dopravních dat, jejíž vizualizací se tato kapitola zabývá, je typ vozidla. Vozidla lze kategorizovat několika způsoby. Otevřená dopravní data Plzeňského kraje [8] vozidla kategorizují do deseti kategorií, které se dále kategorizují do čtyř zjednodušených kategorií, a to motocykly, osobní vozidla, dodávky, nákladní vozidla a autobusy. Takováto kategorická data lze vizualizovat klasicky sloupcovým grafem nebo u zjednodušené kategorizace lze použít i graf výsečový, vzhledem k nízkému počtu kategorií. Ukázka vizualizace typu vozidel výsečovým grafem je na *Obrázku 2.7.*

2.3 Vizualizační nástroje pro tvorbu grafů

Máme-li připravená data a rozmyšleno, jaký typ vizualizace je nejvhodnější, musí se daná vizualizace nějak realizovat, k čemuž existuje několik nástrojů. Od nástrojů, poskytující jednoduché vizualizace v tabulce, až po komplexní 3D vizualizační nástroje. Tato kapitola se zabývá vybranými JavaScriptovými knihovnami umožňující tvorbu a vykreslení grafů, konkrétně Google Charts v *Kapitole 2.3.1* a Charts.js v *Kapitole 2.3.2*. Hlavními kritérii pro výběr knihoven byla kvalita grafického výstupu, kvalita dokumentace, poskytnutí základních typů grafů a rozsáhlost, jelikož některé vizualizační knihovny, jako D3.js[1], jsou pro tuto práci zbytečně komplexní. U obou vybraných knihoven je znázorněno jejich základní použití a ukázka grafického výstupu nad stejnými daty.

2.3.1 Google Charts

Google charts poskytuje několik desítek druhů grafů, mezi kterými samozřejmě nechybí základní typy, jako graf sloupcový, bodový a výsečový. Knihovna je založena na HTML5 (HyperText Markup Language – Hypertextový značkovací jazyk) a SVG (Scalable Vector Graphics – Škálovatelná vektorová grafika) technologii. Poskytnuté grafy jsou interaktivní, dají se přizpůsobovat a navíc lze jednoduše měnit typ grafu, takže pokud se ze začátku rozhodneme, že použijeme například graf výsečový, a poté uvážíme, že by byl lepší graf sloupcový, stačí změnit jednu řádku kódu. Google charts je volně k dispozici i pro komerční účely [9].

[1] https://d3js.org/

Pro používání knihovny je nutno použít soubor, který se stará o načítání jednotlivých grafů, z adresy `https://www.gstatic.com/charts/loader.js`. V podmínkách používání je uvedeno, že je zakázáno provozovat JavaScriptové soubory Googlu na vlastním serveru [10], tudíž lze načtení provést pouze vložením HTML značky, která je popsaná v *Kódu 2.1,* například do hlavičky HTML souboru.

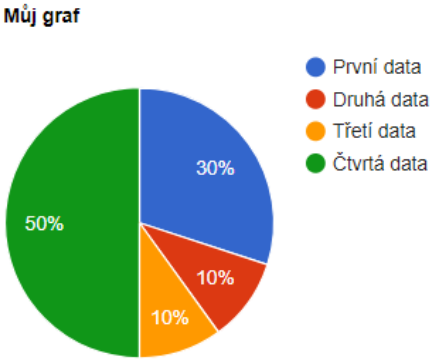
```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
```

Kód 2.1: Načtení Google Charts knihovny

Po načtení souboru `loader.js` se musí před používáním knihovny nadefinovat, jaké typy grafů chceme načíst. Vystačíme-li si se základními typy grafů (sloupcový, spojnicový, výsečový, atd.), stačí načíst balíček `corechart`. Po načtení balíčku již lze vytvořit a naplnit datovou strukturu představující zobrazovaná data, vytvořit příslušný graf a volitelně lze i nastavit vlastnosti grafu. Následně může být graf s danými daty a případnými vlastnostmi zobrazen. Celý postup je znázorněn v *Kódu 2.2*, kde je znázorněno vytvoření výsečového grafu. Grafický výstup *Kódu 2.2* je potom zobrazen na *Obrázku 2.8*.

```
function drawChart() {

    /* Vytvoření dat */
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Název');
    data.addColumn('number', 'Počet');
    data.addRows([
        ['První data', 3],
        ['Druhá data', 1],
        ['Třetí data', 1],
        ['Čtvrtá data', 5]
    ]);

    /* Vytvoření vlastností grafu (titulku, šířky a výšky) */
    var options = {'title': 'Můj graf',
        'width': 400,
        'height': 300
    };

    /* Vytvoření a vykreslení grafu (v předaném HTML elementu s ID 'chart_div') */
    var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
    chart.draw(data, options);
}

/* Načtení balíčku se základními grafy a nastavení, jaká funkce se má po načtení zavolat (tzv.
callback) */
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);
```

Kód 2.2: Vytvoření a vykreslení výsečového grafu v Google Charts



Obrázek 2.8: Grafický výstup Kódu 2.2

2.3.2 Chart.js

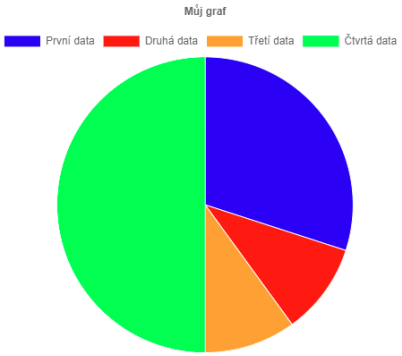
Chart.js poskytuje osm základních druhů grafů, kde jsou všechny animované, přizpůsobitelné a responzivní [11]. Pro renderování grafu používá HTML5 canvas a pro podporu se staršími prohlížeči využívá tzv. *polyfill*, což je typicky JavaScriptový kód, který pro nepodporovaný prohlížeč automaticky doplní určitou funkčnost, aniž by programátor musel něco dělat [12]. Chart.js je open-source projekt pod MIT (Massachusetts Institute of Technology - Massachusettský technický institut) licencí, tudíž jej lze použít i pro komerční účely [13].

Knihovnu, na rozdíl od Google Charts, lze provozovat i na vlastním serveru. Po načtení knihovny a vytvoření příslušného HTML5 canvasu lze začít s vlastním tvořením grafu. Použití knihovny je znázorněno v *Kódu 2.3*, kde je opět vytvořen výsečový graf, který je vyobrazený na *Obrázku 2.9*.

```
<!-- Element, kde bude graf vykreslen -->
<canvas id="mujGraf" width="500" height="400"></canvas>

<!-- Skript pro vytvoření grafu -->
<script>
    var myChart = new Chart(document.getElementById("mujGraf"), {
        /* Typ grafu (zde výsečový) */
        type: 'pie',

        /* Definice dat */
        data: {
            /* Názvy kategorií */
            labels: ["První data",
                "Druhá data",
                "Třetí data",
                "Čtvrtá data"],

            /* Data a jejich grafické vlastností (barvy výplní/ohraničení) */
            datasets: [{
                data: [3, 1, 1, 5],
                backgroundColor: [
                    'rgba(0, 0, 255, 1)',
                    'rgba(255, 0, 0, 1)',
                    'rgba(255, 160, 0, 1)',
                    'rgba(0, 255, 0, 1)',
                ],
```

```
                borderColor: [
                    'rgba(255, 255, 255, 1)',
                    'rgba(255, 255, 255, 1)',
                    'rgba(255, 255, 255, 1)',
                    'rgba(255, 255, 255, 1)',
                ],
            }],

            /* Vlastnosti grafu */
            options: {
                title: {
                    display: true,
                    text: 'Můj graf'
                },
                responsive: false
            }
        }
    });
</script>
```

Kód 2.3 – Vytvoření a vykreslení výsečového grafu v Charts.js

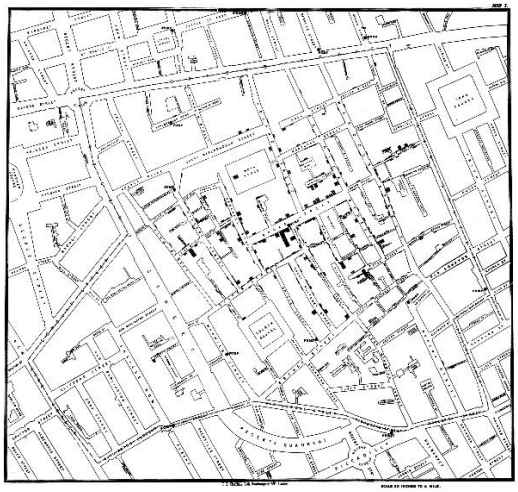


Obrázek 2.9: Grafický výstup Kódu 2.3

3 Vizualizace dat v online mapových portálech

3.1 Vizualizace dat na mapě

V dnešní době je vizualizace dat na mapě oblíbený způsob, jak vizualizovat data spjatá s geografickou polohou, jelikož je to efektní a často i efektivní způsob vizualizace. Jedním z nejznámějších použití vizualizace na mapě pochází z roku 1854, kdy bylo z vizualizace na mapě, vytvořené panem Johnem Snowem, zjištěno, že příčinou vypuknutí cholery v Londýně je kontaminovaná voda z jedné vodní pumpy [3], viz *Obrázek 3.1.*

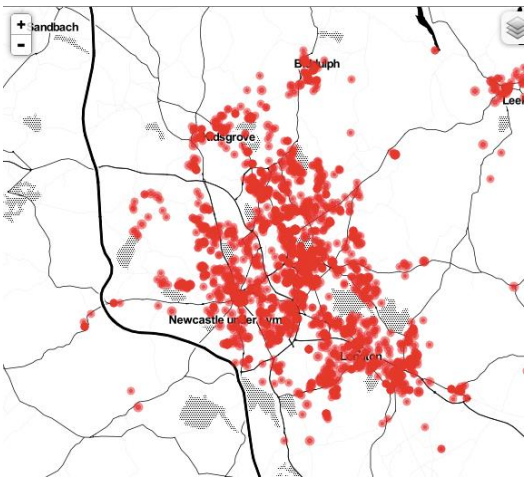


Obrázek 3.1: Mapa vytvořená Johnem Snowem zobrazující místa nakažená cholerou.
Zdroj:
https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak#/media/File:Snow-cholera-map-1.jpg

Vizualizovat data na mapě lze několika způsoby. Data se mohou transformovat například na tvary, jako body a cesty nebo na takzvanou heatmapu, neboli teplotní mapu. Dané tvary nebo heatmapu lze poté nanést na mapu [4]. Existují i další možnosti, které jsou ale mimo téma práce.

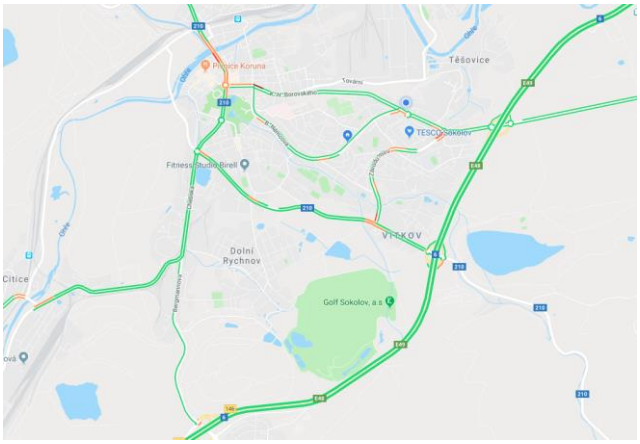
Nejjednodušším způsobem vizualizace dat na mapě je vizualizace pomocí bodů, což znamená, že každý soubor dat se transformuje na bod s danou pozicí a může být navíc doplněn specifickou barvou nebo velikostí, čímž lze znázornit hodnotu daného bodu. Tento způsob je vhodný, pokud se snažíme zdůraznit distribuci nebo shlukování dat, nikoliv však pro porovnání, či kompozici, kde je vhodnější využít jiných vizualizačních

prostředků, jako je například sloupcový graf, či výsečový graf [4], viz *Kapitola 2.1.5*. Příklad vizualizace pomocí bodů lze vidět na *Obrázku 3.2*.



Obrázek 3.2: Mapa Davida Elkse zobrazující kriminální případy v Anglii.
Zdroj: `https://onlinejournalismblog.files.wordpress.com/2015/08/crime-map-stoke.png`

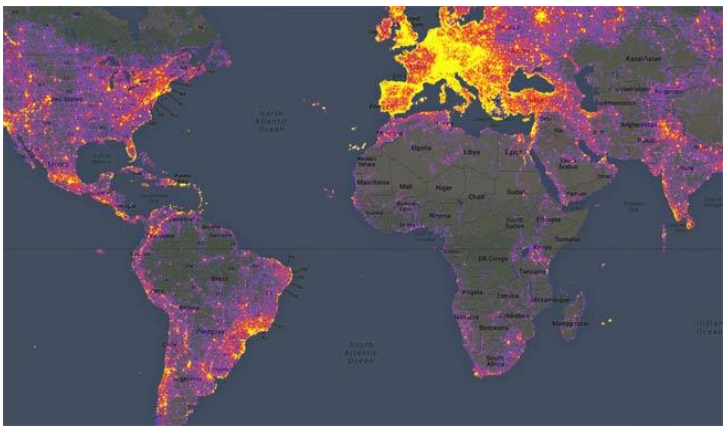
Dále lze data transformovat i na nějaký komplexnější tvar, než jen pouhý bod, například na křivku reprezentující intenzitu provozu na dané silnici, viz *Obrázek 3.3*, nebo na polygon představující četnost populace jednoho státu. Takováto vizualizace je vhodná, pokud se datový soubor nevztahuje k jedné specifické lokaci, ale k širším zeměpisným oblastem, jako například právě k státům [4].



Obrázek 3.3: Vizualizace na mapě, kde jsou barevně zvýrazněné silnice podle intenzity dopravy.
Zdroj: `https://www.google.com/maps`

Posledním zde zmíněným typem vizualizace na mapě je heatmapa, což je pojem, který pochází z roku 1991 od softwarového designéra Cormaca Kinneyho [5]. Heatmapa je grafická reprezentace, kde jsou data transformována do matice obsahující hodnoty představující barvu z určitého spojitého barevného spektra, které začíná na studené barvě, například modré a končí na barvě teplé, například červené [20]. Příklad heatmapy

lze vidět na *Obrázku 3.4*. Heatmapa se nepoužívá pouze jako vrstva, překrývající například mapu světa, ale hojně se používá i ve webové analytice, pro zobrazení chování návštěvníka webu a následné optimalizaci designu webové stránky, viz například nástroj Hotjar[1].



Obrázek 3.4: Příklad heatmapy zobrazující nejvíce navštěvovaná a focená místa.
Zdroj: `https://petapixel.com/assets/uploads/2012/01/heatmap.jpg`

3.2 Mapové zobrazení

Pojmem mapové zobrazení se rozumí způsob zobrazení elipsoidu či koule, například Země, do roviny. Kouli nelze zobrazit do roviny, aniž by se zkreslila plocha nebo úhly. Musí se tedy zvolit, zda budou zkresleny úhly, plocha nebo lze zvolit kompromis, kde se mírně zkreslí jak úhly, tak plocha [14].

Existuje několik typů rozdělení mapového zobrazení. Jedním z nich je rozdělení podle kartografického zkreslení, které mapové zobrazení rozděluje do celkem čtyř kategorií [14]:

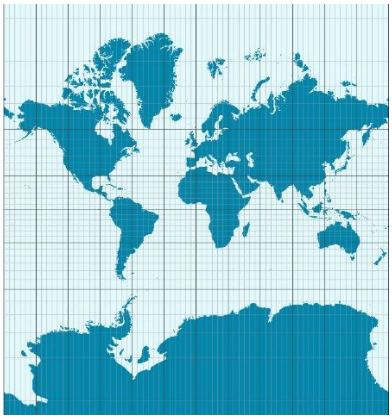
- *Ekvidistantní* - nezkreslují se vzdálenosti v určitém směru
- *Ekvivalentní* - zachovávají se poměry ploch, jsou však zkresleny úhly
- *Konformní* - zachovávají se úhly, jsou však zkresleny plochy
- *Vyrovnávací* - mírně se zkreslují jak úhly, tak plochy

[1] https://www.hotjar.com/

3.2.1 Mercatorovo zobrazení

Jedním z konformních mapových zobrazení je Mercatorovo zobrazení. Příklad mapy vzniklé pomocí Mercatorovo zobrazení lze vidět na *Obrázku 3.5*. Na obrázku je horizontálními čarami naznačeno, že čím je plocha blíže k pólům, tím je více zkreslena.

Toto zobrazení a jeho variace využívá většina mapových portálů zobrazující celý svět, jako například Google Maps [15].



Obrázek 3.5: Mapa vzniklá Mercatorovým zobrazením.
Zdroj: `https://upload.wikimedia.org/wikipedia/commons/0/00/Mercator-proj.jpg`

3.3 Online zobrazení mapy

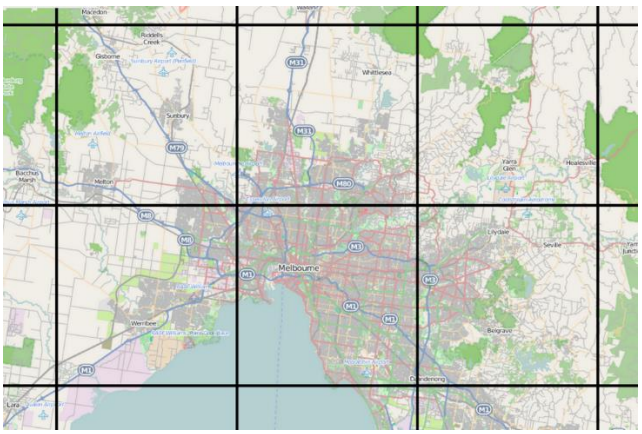
Máme-li již vytvořenou mapu, například pomocí Mercetarovo zobrazení, musí se daná mapa nějakým způsobem zobrazovat. Jedním ze způsobů je zobrazit celou mapu naráz jako jeden obrázek, což ale u větších map, například mapy celé Země, je vysoce neefektivní, vzhledem k tomu, že by obrázek musel mít enormní rozlišení, abychom si mohli detailněji přiblížit nějakou část mapy. U větších map se tedy musí zvolit efektivnější způsob zobrazení.

Jedním z efektivnějších a dnes hojně využívaných způsobů online zobrazení mapy je metoda anglicky pojmenovaná Slippy map (někdy také Tiled web map nebo Raster tile map) [16].

3.3.1 Slippy map

Principem metody je rozdělení mapy na dlaždice a následné stažení a zobrazení pouze těch obrázků (dlaždic), které jsou relevantní pro zobrazení části mapy, kterou si uživatel vyžádá. Dlaždice jsou většinou rastrové obrázky, které vznikají z vektorových

dat. Dlaždice se z výkonnostních důvodů typicky nevytvářejí v reálném čase, ale jsou již vytvořené na serveru [16]. Příklad zobrazené části mapy je na *Obrázku 3.6*.



Obrázek 3.6: Online zobrazení mapy pomocí Slippy map.
Zdroj:
`https://en.wikipedia.org/wiki/Tiled_web_map#/media/File:Tiled_web_map_Stevage.png`

Tato metoda má několik zásadních výhod. Jednou z nich je, že pokud uživatel změní část mapy, kterou chce zobrazit, tak často několik z již stažených dlaždic zůstane stále relevantních, takže se stáhnou jen chybějící dlaždice a poté se správně přeskupí v mřížce.

3.4 Online mapové portály

V dnešní době existuje velké množství statických a interaktivních online mapových portálů nabízejících různé služby pod různými licencemi.

Některé mapové portály také poskytují API (Application Programming Interface – rozhraní pro programování aplikací), což umožňuje programátorovi vkládat do aplikací mapu z daného portálu a provádět s ní různé akce. Mezi základní akce patří zobrazení daného místa a přidání značek na určené místo. Pokročilejší portály poskytují i možnost různého zobrazení mapy, překrytí datovou vrstvou, uživatelskou interakci s mapou, atd.

V dalších kapitolách jsou popsány vybrané nekomerční mapové portály, které bezplatně poskytují API. Konkrétně zahraniční mapový portál Google maps v *Kapitole 3.5* a český mapový portál Mapy.cz v *Kapitole 3.6*. Mezi další vhodné mapové portály, které zde ale popsány nejsou, patří například OpenStreetMap[1]. Hlavními kritérii pro výběr mapových portálů byla jejich univerzálnost, tedy že se nezaměřují na specifickou oblast, dále možnosti zobrazování vlastních dat a kvalita dokumentace.

[1] https://www.openstreetmap.org

3.5 Google maps

Google maps je online mapový portál vyvíjený společností Google. Aplikace byla nasazena v únoru roku 2005. Google maps začal jako C++ desktopová aplikace vyvíjená dánskými bratry Lars a Jens Eilstrup Rasmussen z Where 2 Technologies. Projekt byl poté roku 2004 převzat společností Google a z desktopové aplikace se stala aplikace webová [22].

Google maps využívá mapu vzniklou pomocí Mercatorovo zobrazení a mapa je zobrazena metodou Tiled web map. Aplikace je napsána výhradně v jazyce JavaScript a pro přenos dat využívá formát JSON (JavaScript Object Notation – JavaScriptový objektový zápis) [21].

3.5.1 API

V červnu roku 2005 bylo zveřejněno Google Maps API, což umožňuje vložení Google Maps do jiných aplikací. K dnešnímu dni je vydáno několik verzí Google Maps API, jako například verze pro Android, iOS a JavaScriptové API [21]. Dále se předpokládá, že se využívá pouze JavaScriptové API.

Google Maps API je možno volně používat, dokud se nepřekročí limit 25 000 tisíc zobrazení za 24 hodin. Pokud se předpokládá, že limit bude překračován, je možné aktivovat službu, která po překročení limitu limit na den za poplatek navýší [17].

Pro používání Google Maps API je nutno získat API klíč. Pro jeho získání je nutné nejdříve zaregistrovat aplikaci na Google API Console, kde se následně klíč vygeneruje. Je-li aplikace již zaregistrována a klíč vygenerován, lze začít Google Maps API v aplikaci využívat [21]. Vložení mapy do aplikace je pak už jednoduché. Stačí vložit do HTML těla například *Kód 3.1.*

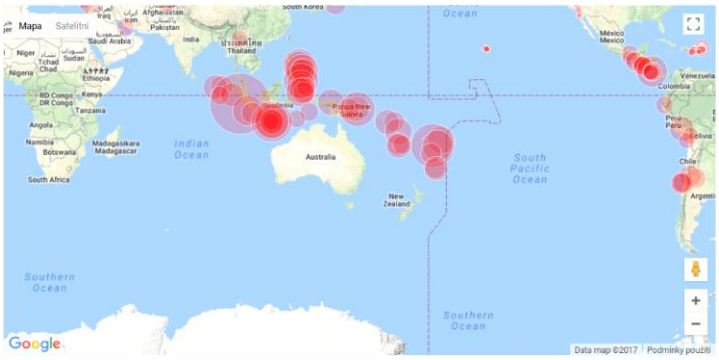
```
<div id="map"></div>
<script>
    let map;
    function initMap() {
        map = new google.maps.Map(document.getElementById('map'),
            {
                center: {lat: -34.397, lng: 150.644},
                zoom: 15
            });
    }
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=API_KEY&callback=initMap" async
defer></script>
```

Kód 3.1: Vložení Google mapy do aplikace

Pokud se `API_KEY` v *Kódu 3.1* nahradí API klíčem aplikace, tak daný skript zajistí vykreslení mapy, která je vycentrovaná na Sydney, Austrálie s viditelnými ulicemi. To, že mapa bude vycentrovaná na Sydney, zajišťuje parametr `center`, který představuje souřadnice středu vykreslené mapy. Nastavením parametru `zoom` se určuje přiblížení mapy. Následující seznam obsahuje očekávané úrovně detailu při různých nastaveních parametru zoom [21]:

- 1 – Svět
- 5 – Kontinenty
- 10 – Města
- 15 – Ulice
- 20 - Budovy

Do vytvořené mapy lze navíc přidat například značky, různé tvary nebo lze mapu překrýt datovou vrstvou, jako například heatmapou, viz *Kapitola 3.1*. Příklad mapy s vykreslenými tvary je znázorněn na *Obrázku 3.7*.



Obrázek 3.7: Google mapa s vykreslenými tvary představující sílu zemětřesení
Zdroj: `https://developers.google.com/maps/documentation/javascript/earthquakes`

3.6 Mapy.cz

Mapy.cz je mapový portál spravovaný společností Seznam.cz. Aplikace byla spuštěna roku 1998, kdy jí ještě provozovala firma PJsoft s.r.o. a obsahovala pouze automapu České republiky a ulice vybraných měst. Roku 2005 přišel seznam s inovovanou verzí, která již zobrazovala podrobně města po celém území Česka. Tuto verzi si Seznam.cz již spravoval sám [23]. Roku 2015 byla vypuštěna verze obsahující mapu celého světa a přešla také na Mercatorovo zobrazení, jako ostatní celosvětové portály. Celosvětová data

čerpá z OpenStreetMap, což je otevřený projekt, jehož cílem je tvorba volně dostupných geografických dat [24].

Portál nabízí kromě základní mapy také například mapu turistickou, zimní, leteckou, nebo také mapu historickou, což je historická mapa Čech. Mezi poskytované služby patří plánovač trasy, prohlížení českých měst ve 3D nebo Panorama, což je obdoba Street View portálu Google Maps [23].

3.6.1 API

API bylo společností Seznam.cz uvolněno roku 2006 a je možno ho používat zcela zdarma i pro komerční účely. Na rozdíl od Google Maps API lze Mapy.cz API ihned začít používat bez nutnosti aplikaci registrovat a získání API klíče [18]. Příklad vložení mapy do aplikace je uveden v *Kódu 3.2*.

```
<head>
    <script src="https://api.mapy.cz/loader.js"></script>
    <script>Loader.load()</script>
</head>

<body>
<div id="mapa" style="width:600px; height:400px;"></div>
<script type="text/javascript">
    var stred = SMap.Coords.fromWGS84(14.41, 50.08);
    var mapa = new SMap(JAK.gel("mapa"), stred, 10);
    mapa.addDefaultLayer(SMap.DEF_BASE).enable();
    mapa.addDefaultControls();
</script>
</body>
```

Kód 3.2: Vložení Mapy.cz do aplikace

API také poskytuje základní funkce pro překrytí mapy datovou vrstvou a pro její manipulaci, ale jejich počet je ve srovnání s Google Maps API výrazně menší.

4 Návrh

Ve specifikaci nebyly uvedeny limity na použité technologie, ani jak má být aplikace pojata. Pouze, že se má jednat o online aplikaci. Aplikace by tedy mohla být navržena několika způsoby. Jednou z možností je, že by aplikace běžela na samostatné stránce ve webovém prohlížeči a mohla by se při jakékoliv akci, například přihlášení, celá znovu načíst. Další variantou, která byla nakonec zvolena, je pojetí aplikace jako komponenty, kterou lze vložit do již existující stránky a „žije si“ v přiděleném prostoru, bez ovlivňování zbytku načtené stránky. Tato komponenta by tedy měla vše zpracovávat bez nutnosti znovunačtení celé stránky. K docílení toho, že aplikace toto omezení neporuší, lze využít AJAX (Asynchronous JavaScript and XML - asynchronní JavaScript a XML), viz *Kapitola 4.3.4*.

4.1 Specifikace požadavků

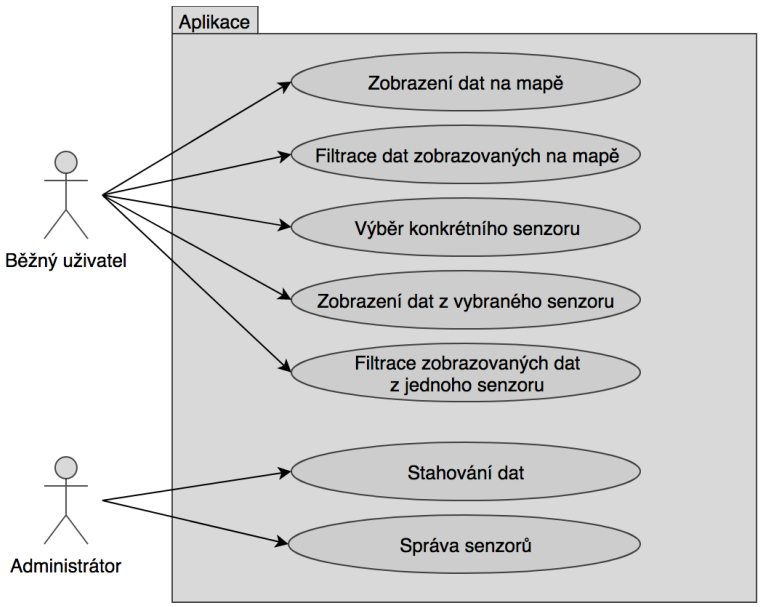
V následujícím seznamu jsou popsány požadavky na výslednou aplikaci:

- Aplikace bude zpracovávat a vizualizovat otevřená silniční data Plzeňského kraje.
- Aplikace bude vizualizovat data online.
- Data se budou vizualizovat v online mapovém portálu a v grafech.
- Uživatel si bude moct vybrat, jaká data se budou vizualizovat.
- Aplikace bude schopna stahovat denní data, která poté budou k dispozici pro vizualizaci.
- Administrátor bude moct jednotlivým senzorům nastavit jejich polohové souřadnice.

4.2 Případy užití

V aplikaci existuje několik případů užití, které se dají rozdělit na případy užití pro běžného uživatele a pro správce aplikace neboli administrátora. Běžný uživatel bude aplikaci používat především pro zobrazení jím vybraných dat, zatímco administrátor se bude starat o správu senzorů a o to, aby byla pro vizualizaci dostupná aktuální data. Na

Obrázku 4.1 je znázorněn diagram případů užití. Jednotlivé případy užití jsou následně popsány.



Obrázek 4.1: Diagram případů užití

- Běžný uživatel
  - *Zobrazení dat na mapě* – Uživatel si bude moct silniční data zobrazit v online mapovém portálu.
  - *Filtrace dat zobrazovaných na mapě* – Uživatel si bude moct navolit, jaká data chce do vizualizace na mapě zahrnout.
  - *Výběr konkrétního senzoru* – Jednotlivé senzory budou vyznačeny na mapě a uživatel si bude moct libovolný z vyznačených senzorů vybrat, čímž se mu umožní zobrazení dat zaznamenaných daným senzorem.
  - *Zobrazení dat z vybraného senzoru* – Uživatel si bude moct, po vybrání konkrétního senzoru, zobrazit data naměřená daným senzorem v grafech.
  - *Filtrace zobrazovaných dat z jednoho senzoru* – Podobně, jako u druhého případu užití, si bude moct uživatel navolit, jaká data budou ve vizualizaci dat z jednoho senzoru zahrnuta.
- Administrátor
  - *Stahování dat* – Administrátor bude moct stahovat data z jednotlivých dnů, která poté budou moct být vizualizována.

- *Správa senzorů* – Vlastnosti jednotlivých senzorů budou moct být změněny administrátorem

4.3 Použité technologie

4.3.1 HTML a CSS

HTML společně s CSS (Cascading Style Sheets - kaskádové styly) patří mezi nejzákladnější technologie pro tvorbu webových stránek a aplikací. Webové prohlížeče podle HTML a CSS souborů, které jim zašle například webový server, rendrují výsledný vzhled stránky. Jazyk HTML je charakterizován množinou tzv. tagů a jejich atributů. Jedněmi z atributů jsou `class` a `id`, které využívá mimo jiné jazyk CSS k popisu vzhledu daných tagů [25].

HTML je v aplikaci použit pro nadefinování šablon a CSS k nadefinování vzhledu aplikace.

4.3.2 JavaScript a JavaScriptové knihovny

JavaScript je interpretovaný programovací jazyk zpravidla používán pro programování klientské části webových aplikací. Kód lze vložit přímo do HTML kódu nebo jej lze mít v zvláštním souboru, který se poté dá do HTML stránky naimportovat. Jak již bylo řečeno, JavaScript se používá především pro programování klientské části, tudíž se většinou spouští až u cílového uživatele, na rozdíl od PHP. Existují ale i implementace, které umožňují používat JavaScript na straně serveru, jako například Node.js [26].

Jednou z použitých JavaScriptových knihoven je knihovna JQuery[1]. JQuery je volně dostupná JavaScriptová knihovna umožňující jednodušší používání JavaScriptu. JQuery samo řeší například meziprohlížečovou kompatibilitu a nabízí různé funkce, jako efekty, animace nebo jednodušší používání AJAXu. JQuery lze do aplikace zakomponovat několika způsoby. Je možno provozovat knihovnu na vlastním serveru nebo ji lze naimportovat z CDN (Content Delivery Network – Síť pro doručování obsahu) například

[1] https://jquery.com/

Googlu. Výhoda tohoto postupu je v tom, že uživatel může mít tuto knihovnu již uloženou v cache paměti prohlížeče, takže se nemusí znovu stahovat.

Dále je použit JQueryUI[1], což je sada postavená nad jQuery poskytující například různé GUI (Graphical User Interface – Grafické uživatelské rozhraní) prvky, efekty a motivy.

JavaScript verze ECMAScript 6, společně s použitými knihovnami, je v aplikaci použit v klientské části aplikace.

4.3.3 Google Maps a Google Charts

Pro zobrazení mapy a dat na ní bylo nakonec zvoleno Google Maps a pro zobrazení grafů Google Charts. K těmto volbám vedlo především intuitivní používání, kvalita dokumentace a kvalita grafického výstupu.

4.3.4 AJAX

AJAX je označení pro množinu webových technologií, umožňující tvorbu asynchronních interaktivních webových aplikací. AJAX není knihovna nebo framework, ale pouze jistý způsob programování s využitím klasických webových technologií. Pomocí AJAXu může klientská část aplikace poslat na server požadavek a přijmout odpověď asynchronně, což umožňuje komunikovat se serverem a měnit obsah stránky bez nutnosti obnovení celé stránky. Písmeno X, představující XML (Extensible Markup Language, rozšiřitelný značkovací jazyk), je v názvu je zavádějící, jelikož AJAX dokáže pracovat i s jinými formáty dat, jako například často používaný JSON, prostý text nebo i binární data [27].

Pomocí AJAXu a formátu JSON je v aplikaci naimplementovaná veškerá komunikace klient – server.

4.3.5 PHP

PHP je dnes nejpoužívanější [19] skriptovací programovací jazyk používaný k tvorbě webových aplikací. V případě použití PHP pro webové aplikace, jsou skripty prováděny na straně serveru a k cílovému uživateli je přenesen pouze výsledek skriptu [28].

[1] https://jqueryui.com/

V aplikaci je v PHP implementována veškerá serverová část, tedy serverová část kontroléru a datová vrstva, viz *Kapitola 4.7*.

4.3.6 MySQL

MySQL je velmi populární, volně dostupný relační systém řízení báze dat. K dotazování využívá, jak již lze z názvu odvodit, jazyk SQL (Structured Query Language, strukturovaný dotazovací jazyk) s některými rozšířeními [29].

MySQL je v aplikaci využito k ukládání perzistentních dat a vykonáváním dotazů nad danými daty.

4.4 Zpracovávaná data

Aplikace zpracovává a vizualizuje otevřená dopravní data Plzeňského kraje, která jsou sbírána v rámci nasazení tzv. PAM (Prvky Aktivního Monitoringu) na území Plzeňského kraje. PAM je výpočetní jednotka s jednou nebo více kamerami, která v dané lokalitě sbírá a zpracovává data. Data jsou dále zpracovávána do dvou kategorií, a to pro potřeby Plzeňského kraje a pro Policii ČR. Data pro Plzeňský kraj jsou anonymizována, nelze tedy přesně zjistit, o jaké vozidlo se jedná. Z dat lze zjistit pouze počet průjezdů typových dopravních prostředků užívaných na pozemních komunikacích v čase, včetně jejich rychlostí. Volně k dispozici jsou data právě pro potřeby Plzeňského kraje, tedy ty, která aplikace zpracovává [8].

4.4.1 Způsob vystavení dat

```
<tr class='file'>
    <td>
        <a href='./DOPR_D_20180401.zip' class='name'>DOPR_D_20180401.zip</a>
    </td>
    <td>
        <a href='./DOPR_D_20180401.zip'>ZIP Archív</a>
    </td>
    <td sorttable_customkey='7875023'>
        <a href='./DOPR_D_20180401.zip'>7.5 MB</a>
    </td>
    <td sorttable_customkey='20180402050229'>
        <a href='./DOPR_D_20180401.zip'>Apr 2 2018 5:02 AM</a>
    </td>
</tr>
```

Kód 4.1: Ukázka vystavení dat z 1.4.2018

Data jsou zveřejňována v denních intervalech na dopravní webové stránce Plzeňského kraje[1] v zip archivu. Databáze, kde jsou data uložena, není veřejně dostupná. K datům lze tedy přistupovat pouze přes výše zmíněnou webovou stránku. Jednotlivá data jsou vystavována, jako řádky HTML tabulky obsahující odkaz na stažení archivu, viz ukázka v *Kódu 4.1*.

4.4.2 Soubory v archivu

Archiv obsahuje celkem tři soubory. Soubor s obecným popisem dat, soubor se senzory a soubor s naměřenými daty danými senzory. Ukázky souboru se senzory a souboru s naměřenými daty jsou v *Příloze A*.

Soubor se senzory je uložen ve formátu CSV (Comma-Separated Values, hodnoty oddělené čárkami) a obsahuje tabulku se seznamem všech dostupných senzorů zaznamenávající silniční data. Tabulka se skládá z následujících pěti sloupců:

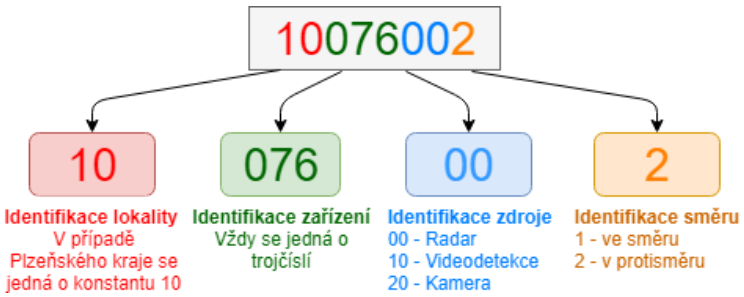
- *Name* - jméno senzoru
- *Town* - město, ve kterém se senzor nachází
- *Street* - ulice, ve které se senzor nachází
- *IdDevice* - id senzoru
- *IdArea* - id oblasti, ve které se senzor nachází

Soubor s daty naměřenými senzory je opět uložen ve formátu CSV a obsahuje tabulku se všemi zaznamenanými vozidly za daný den ze všech senzorů. Jeden záznam tedy představuje právě jedno zaznamenané vozidlo a jejich počet se za jeden den pohybuje zpravidla kolem jednoho milionu. Tabulka obsahuje celkem jedenáct sloupců, kde pro aplikaci jsou relevantní pouze čtyři, a to sloupce následující:

- *IdDetektor* – Osmimístné číslo, které představuje několik informací, viz příklad na *Obrázku 4.2*.
- *DatumCas* – Datum a čas pořízení záznamu.
- *Rychlost* – rychlost zaznamenaného vozidla. Rovná-li se nule, jedná se o chybně naměřenou rychlost.
- *TypVozidla* – Zjednodušené označení typu vozidla, kde možné typy vozidel jsou následující:

[1] http://doprava.plzensky-kraj.cz/opendata/doprava/den/

- 0 – neznámý
- 1 – motocykl
- 2 – dodávka
- 3 – nákladní vozidlo nebo autobus



Obrázek 4.2: Popis sloupce IdDetektor

Jelikož jsou data pořizována senzory v módu založeném na poloze, viz *Kapitola 2.2.1*, součástí dat není polohová složka. Poloha není zmíněna ani v souboru obsahující senzory. Poloha senzorů tedy není součástí otevřených dat a je nutno ji zjistit manuálně, například z dopravní mapy Plzeňského kraje[1].

4.5 Serverová část aplikace

Serverová část aplikace má na starosti několik úloh, mezi které patří stahování otevřených silničních dat Plzeňského kraje, jejich následné zpracování a uložení. Dále poskytnutí přístupu k těmto datům včetně kontroly, zda má uživatel oprávnění k provedení činnosti nad danými daty.

4.5.1 Stahování dat

Stahování otevřených dopravních dat Plzeňského kraje je možno provést několika způsoby.

Jedním ze způsobů je vytvoření skriptu, který by se periodicky spouštěl na serveru a kontroloval by, zda nejsou k dispozici nová data. Pokud by byly, tak by je stáhnul. Skript by se musel přidat do plánovače úloh příslušného operačního systému, jako je *cron* v unixových operačních systémech, či *Task Scheduler* ve Windows. Nevýhodou tohoto způsobu, na rozdíl od způsobu popsaného v dalším odstavci, je komplikovanější

[1] http://doprava.plzensky-kraj.cz/map/index

nasazení aplikace a menší kontrola nad tím, jaká data se budou stahovat. Naopak výhodou je automatizace.

Dalším z možných způsobů je vytvoření administrátorského prostředí, kde by se administrátorovi zobrazil stav dat a mohl by dosud nestáhnuté data případně stáhnout. Výhodou tohoto způsobu je maximální kontrola nad tím kdy a jaká data se stahují, přehlednost pro administrátora a jednodušší nasazení aplikace. Nevýhodou je, že se data musí stahovat manuálně.

Zvolen byl nakonec druhý způsob, tedy stahování dat přes administrátorské prostředí. K této volbě vedlo především jednodušší nasazení aplikace.

4.5.2 Zpracování dat

Mezi základní zpracování dat patří zbavení se nepotřebných sloupců, aby zbyly pouze ty sloupce, které jsou relevantní pro aplikaci.

Vzhledem k tomu, že se počet záznamů během jednoho dne pohybuje zpravidla kolem jednoho milionu, byla zvážena i agregace dat. Výsledky v *Kapitole 6*.1 ukázaly, že se díky agregaci, v případě této aplikace, několikanásobně zlepší doba odezvy za cenu přijatelné ztráty informace. Agregace je tedy provedena. Data se agregují takovým způsobem, aby se ztratilo co možná nejméně informace. Nejdříve je nutno se rozhodnout, v jakém časovém intervalu se bude agregace provádět. Po zvážení byla vybrána agregace v patnáctiminutových intervalech, jelikož větší interval by již nemusel přesně odrážet vlastnosti dopravy na dané silniční komunikaci a menší interval je již zbytečně jemný.

Jedním z možných způsobů agregace je následující – za daný časový interval se sečte počet jednotlivých typů vozidel zaznamenaných jedním senzorem v jednom směru a zprůměruje se jejich rychlost. Je také vhodné si uchovávat maximální a minimální rychlost daných typů vozidel za daný časový interval. Součástí agregovaných dat za jeden časový interval jsou tedy následující složky:

1. datum a čas počátku intervalu
2. délka intervalu
3. id senzoru, kterým byla data zaznamenána
4. směr, ve kterém byla vozidla zaznamenána

5. počet správě zaznamenaných vozidel daného typu vozidla (správně zaznamenaným vozidlem je myšleno zaznamenané vozidlo, u něhož byla správně zaznamenána rychlost)
6. počet chybně zaznamenaných vozidel daného typu vozidla
7. průměrná rychlost daného typu vozidla
8. maximální rychlost daného typu vozidla
9. minimální rychlost daného typu vozidla

Jelikož jsou celkem čtyři typy vozidel a navíc neznámý typ vozidla, jsou složky 5. – 9. v agregovaných datech celkem pětkrát – jednou pro každý typ vozidla.

4.5.3 Uchování dat

K uchování dat byl zvolen relační systém řízení báze dat MySQL, jelikož se jedná o standartní způsob pro uchovávání objemných dat a je volně dostupný k používání, viz *Kapitola 4.3.6*. Tímto způsobem jsou uložena všechna persistentní data, tedy informace o senzorech, agregovaná data ze senzorů a uživatelé, viz *Kapitola 4.5.4*.

Senzory mohou být do databáze vkládány téměř jedna ku jedné k obsahu staženého souboru obsahující senzory. Součástí řádku v databázi, představující jeden senzor, mohou být tedy všechny sloupce popsané v odstavci *Kapitoly 4.4.2* pojednávajícím o souboru se senzory. Pouze tyto sloupce ale nejsou dostačující, jelikož je nutné k senzorům dodat jejich polohové souřadnice a souřadnice cest, viz *Kapitola 4.4.2*.

Data pořízena senzory nelze ukládat jedna ku jedné k obsahu souboru obsahující záznamy ze senzorů, jak to je provedeno u senzorů, jelikož je prováděna agregace. Součástí řádku v databázi by mělo být všech 29 složek popsaných v odstavci předchozí kapitoly, kde je popsána agregace dat.

4.5.4 Přístup k datům

Pro přístup k datům je vhodné implementovat vrstvu zajišťující vkládání, úpravu a čtení dat z databáze. Vrstva by dále měla zajišťovat případné dodatečné zpracování dat do formátu vhodného pro klientskou část aplikace.

Je také nutné nějakým způsobem zajistit, aby všichni uživatelé nemohli provádět určité akce, například úpravu senzorů, či agregovaných dat v databázi, k čemuž lze

vytvořit jednoduchý přihlašovací systém. Pro přihlašovací systém lze vytvořit v databázi tabulku uživatelů s jejich uživatelským jménem, heslem a rolí. Pod pojmem role se myslí například administrátor. V aplikaci bude zatím zahrnuta pouze administrátorská role, jelikož vytvoření dalších rolí prozatím postrádá smysl.

4.6 Klientská část aplikace

Klientská část aplikace neboli ta část, která běží ve webovém prohlížeči na straně klienta, se stará zejména o zobrazování dat. Součástí aplikace bude i administrátorské prostředí, které slouží pro stahování dat. Další vhodnou funkcionalitou pro administrátorské rozhraní je správa senzorů. Klientská část se tedy dělí na administrátorskou část a část pro nepřihlášeného uživatele.

4.6.1 Nepřihlášený uživatel

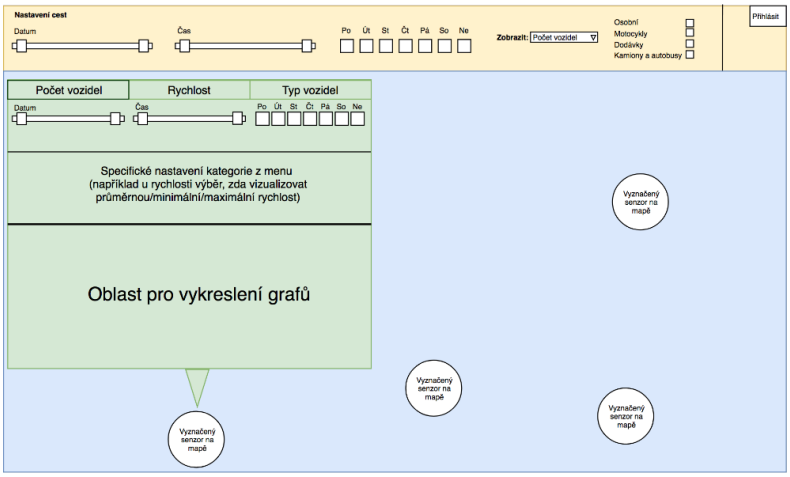
Část pro nepřihlášeného uživatele slouží pouze pro vizualizaci dat. V dalších odstavcích je popsán návrh vizualizace a GUI.

Jelikož se vizualizují silniční data, což jsou data spjatá s geografickou polohou, je vhodné využít vizualizace na mapě. Silniční data, která jsou snímána senzory v módu založenými na poloze, lze na mapě interaktivně vizualizovat několika způsoby. Například vizualizací bodovou či čárovou, viz *Kapitola 2.2.3*. Zvolena byla čárová vizualizace z důvodu lepší přehlednosti pro uživatele. Význam čar je možné specifikovat. Mezi vhodné významy čar, vzhledem ke zpracovávaným datům, patří počet průjezdů vozidel a průměrná, maximální a minimální rychlost. Dále je možné nastavit minimální a maximální datum a čas, z jejichž rozmezí se data vizualizují. Mezi další vhodné nastavení patří možnost výběru, jaké dny v týdnu a jaké typy vozidel budou ve vizualizaci zahrnuty. Nakonec je nutno vyřešit, jakým způsobem se budou čáry barevně škálovat. Byla zvolena barevná škála od zelené barvy k červené, a to rozdílným způsobem pro vizualizaci počtu vozidel a rychlostí. Barva se u počtu vozidel škáluje relativně od nejméně frekventovaného místa k nejvíce frekventovanému místu, zato u rychlosti se škálují mezi pevně danými konstantami.

Další vhodnou vizualizací je využití grafu. Grafem se vizualizují agregovaná data naměřená z jednoho senzoru a lze z nich vyčíst, jak se mění počet vozidel a rychlost

v čase a poměr zaznamenaných typů vozidel. Pro vizualizaci počtu vozidel a rychlostí byl zvolen sloupcový graf, kde lze mimo jiné navolit, zda se bude vizualizovat podle času, dnů v týdnu, měsíců či roků a jaké typy vozidel se mají ve vizualizaci zahrnout. Dále je vizualizován poměr mezi zaznamenanými typy vozidel, na což je vhodný výsečový graf. U všech grafů je opět možné navolit rozmezí dat a času, zahrnuté dny v týdnu a navíc, zda se má rozlišovat směr, tedy jestli se má pro oba směry vykreslit zvláštní graf nebo zda se mají oba směry sloučit do jedné datové řady.

Návrh GUI pro nepřihlášeného uživatele je znázorněno na *Obrázku 4.3*, kde je žlutě zvýrazněn panel pro nastavení zobrazení cest a přihlášení se do administrátorského prostředí. Modře je zvýrazněna část, kde je vykreslena mapa a na ní kruhy představující senzory, u nichž budou vykresleny cesty. Po kliknutí na kolečko představující senzor se u daného senzoru otevře okno, které je v návrhu zvýrazněno zeleně. V okně lze nastavit a vykreslit jednotlivé grafy.



Obrázek 4.3: Návrh GUI pro nepřihlášeného uživatele

4.6.2 Administrátor

Administrátorská část slouží k nastavení některých parametrů senzorů a k stahování nových dat.

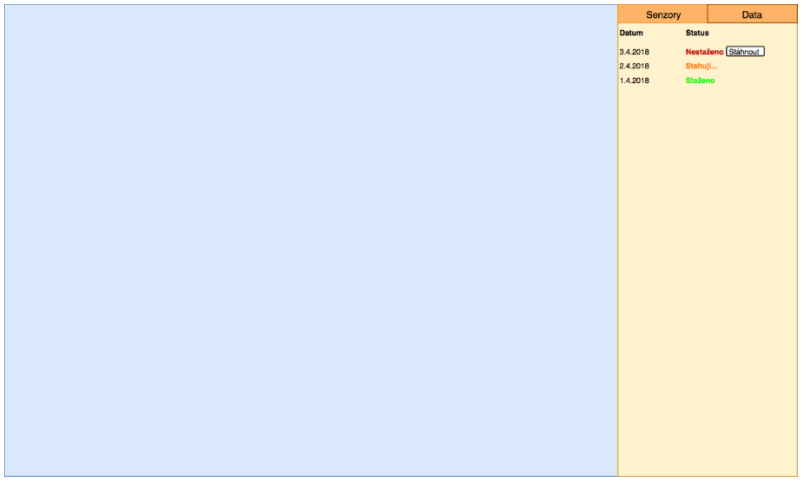
Mezi nastavitelné parametry senzorů patří hlavně souřadnice senzoru a souřadnice cest ve směru a v protisměru k senzoru. Nastavení těchto souřadnic by šlo provést například poskytnutím textových polí pro souřadnice senzoru a souřadnice obou cest, do kterých by administrátor souřadnice vepsal, což by ale nebylo příliš pohodlné. Lepším

způsobem je opět využití mapy, na které by bylo možné intuitivně vybrat jak souřadnice senzoru, tak souřadnice cest.

Pro stahování dat je vhodné vytvořit seznam, ve kterém budou vypsány všechny dny, u kterých jsou k dispozici naměřená data. U všech dnů by měl být vypsán status – zda jsou data již stažená, či nejsou. Pokud by data stažena nebyla, mělo by u daného dne být k dispozici tlačítko pro případně stažení.

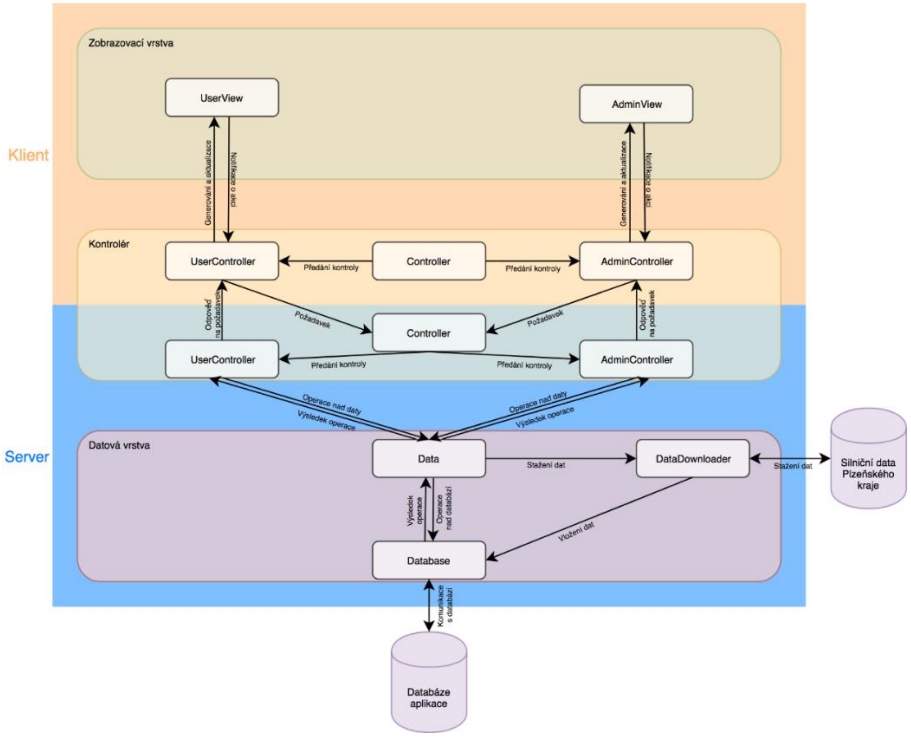
Návrh administrátorského GUI je znázorněn na *Obrázku 4.4* a *Obrázku 4.5*. Na prvním obrázku je znázorněna správa senzorů, kde je žlutě zvýrazněn panel obsahující seznam senzorů společně se statusem nastavení souřadnic a cest. Panel dále obsahuje nastavení právě vybraného senzoru. Modře je zvýrazněn prostor představující mapu, na které je vykreslen právě vybraný senzor a jeho cesty. Na druhém obrázku je znázorněno stahování dat.

Obrázek 4.4: Návrh GUI pro administrátora – správa senzorů



Obrázek 4.5: Návrh GUI pro administrátora - stažení dat

4.7 Architektura aplikace



Obrázek 4.6: Návrh architektury aplikace

Na *Obrázku 4.5* je znázorněn návrh architektury aplikace. Jak je vidět, aplikace je rozdělena celkem do tří vrstev – zobrazovací vrstvy, kontroléru a datové vrstvy.

Zobrazovací vrstva se stará především o zobrazení GUI prvků, zobrazovací logiku a případné rychlé zpracování dat do vhodné podoby, jako je konverze škály na barvu. Dále vrstva poskytuje informace o tom, jaké hodnoty jsou navoleny na daných nastavovacích prvcích. V neposlední řadě umožňuje kontroléru k danému tlačítku přidat obslužnou akci.

Datová vrstva má na starosti veškeré činnosti týkající se persistentních dat. Poskytuje tedy čtení, vkládání a modifikaci dat v databázi a v případě čtení i následné zpracování dat do formátu vhodného pro klientskou část. Datová vrstva se stará i o činnosti, které vyžadují komunikaci s webovou stránkou obsahující silniční data. Mezi takovéto činnosti patří zjišťování, z jakých dnů jsou data k dispozici a jejich případné stažení, zpracování a následné vložení do databáze.

Kontrolér je vrstva nacházející se mezi zobrazovací a datovou vrstvou. Tato vrstva je rozdělena na dvě části – klientskou a serverovou. Tyto dvě části spolu navzájem komunikují nejčastěji takovým způsobem, že klientská část zašle serverové části

požadavek na provedení nějaké operace či na navrácení určitých dat, na který serverová část odpoví výsledkem operace, případně požadovanými daty.

Klientský kontrolér se kromě komunikace se serverovým kontrolérem stará o obsluhu tlačítek ze zobrazovací vrstvy, které slouží k provedení akce vyžadující komunikaci se serverem. Dále se vrstva stará o modifikaci GUI přes zobrazovací vrstvu a o adekvátní reakci na případné chyby.

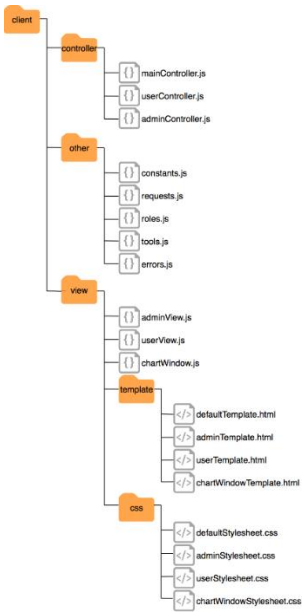
Serverový kontrolér se stará o zpracovávání požadavků klientského kontroléru. Před samotným zpracováním serverový kontrolér nejdříve kontroluje, zda uživatel, od kterého požadavek přišel, má příslušné práva k realizaci daného požadavku. Pokud kontrola proběhne úspěšně, přistupuje se ke čtení, modifikaci či vkládání dat přes datovou vrstvu a následné navrácení dat, či výsledku operace zpět klientskému kontroléru.

5 Realizace

Zdrojový kód aplikace je rozdělen do dvou částí – klientské a serverové. Na úrovni těchto částí se navíc nachází JavaScriptový modul `trafficVisualisation`, viz dále, který slouží jako vstupní bod do aplikace. Tento modul poskytuje funkci pro inicializaci aplikace. Inicializační funkce vyžaduje obalový HTML prvek pro aplikaci a API Google Maps API klíč. Modul dále zajišťuje načtení všech použitých knihoven, API, kaskádových stylů a ostatních modulů klientské části. Po načtení všeho potřebného je předána kontrola hlavnímu klientskému kontroléru aplikace `mainController`, který je podrobněji popsán v *Kapitole 5.1.1*.

Součástí aplikace je také konfigurační soubor, kde lze nastavit konfigurovatelné hodnoty, mezi které patří například přihlašovací údaje k databázi. Navíc lze nastavit parametry struktury stahovaných dat pro případ, že by se struktura stahovaných dat v budoucnu změnila. Všechny hodnoty jsou ve výchozím stavu souboru zakomentovány a mají nastavenou výchozí hodnotu, která je nadefinována v aplikaci.

5.1 Klientská část



Obrázek 5.1: Adresářová struktura klientské části

Klientská část je implementována především v jazyce JavaScript s využitím tzv. *Revealing Module Pattern*, což je způsob, jakým lze aplikaci rozdělit do modulů, čímž lze

docílit rozdělení kódu do souvisejících částí kódu. V aplikaci platí, že každý JavaScriptový soubor obsahuje právě jeden modul a název daného modulu je totožný s názvem souboru, ve kterém se modul nachází. Aplikace je dále dělena do jmenných prostorů, kde základním jmenným prostorem je `trafficVisualisation`, který se dále dělí na `controller`, `view` a `other`. Struktura jmenných prostorů se shoduje s adresářovou strukturou klientské části aplikace, která je znázorněna na *Obrázku 5.1*, kde jsou mimo jiné také znázorněny všechny moduly aplikace, šablony a soubory s kaskádovými styly. Jednotlivé moduly jsou podrobně popsány v následujících podkapitolách.

5.1.1 Modul `mainController`

Tomuto modulu je předána kontrola po nahrání všech potřebných souborů pro aplikaci modulem `trafficVisualisation`. Modul si nejdříve pomocí AJAXu ze serveru zjistí, jakou má uživatel roli, tedy jestli již není přihlášen jako administrátor a stránka nebyla například pouze obnovena. Po zjištění role předá kontrolu specifickému kontroléru – `userController` či `adminController`. Roli si navíc kontrolér uchovává v proměnné, kterou poskytuje specifickým kontrolérům. Důvod, proč je nutno si současnou roli uchovávat je znázorněn na následujícím příkladu – nepřihlášený uživatel si vyžádá vykreslení cest a ihned poté se přihlásí jako administrátor. Jelikož jsou požadavky prováděny pomocí AJAXu, tedy asynchronně, může být nejdříve obsloužen méně náročný požadavek, kterým je požadavek na přihlášení. Po obsloužení náročnějšího požadavku, tedy požadavku na vykreslení cest, je již zobrazeno administrátorské rozhraní, kde cesty nelze vykreslit. Oba klientské kontroléry – `userController` a `adminController` – si tedy po přijetí výsledku požadavku kontrolují, zda je nastavena stále stejná role, jaká byla nastavena při odeslání požadavku. Pokud není, tak výsledek požadavku ignoruje. Modul v poslední řadě předává specifickým kontrolérům obslužné funkce pro přihlášení a odhlášení. V poslední řadě modul zajišťuje předávání kontroly mezi kontroléry při změně role.

5.1.2 Modul **userController**

Tento modul je kontrolérem pro nepřihlášeného uživatele. Modul nejdříve volá příslušnou funkci modulu `userView` ze zobrazovací vrstvy, která se stará o inicializaci GUI. Následně přidává tlačítkům vyžadující komunikaci se serverem obslužné funkce, které jsou vykonávány pomocí AJAXu. Modul, podle navrácených dat ze serveru, volá příslušné funkce modulu `userView`, které se starají o aktualizaci GUI.

5.1.3 Modul **adminController**

Tento modul má stejnou funkci, jako modul předchozí s tím rozdílem, že se jedná o administrátorský kontrolér, s kterým je spjatý modul `adminView` ze zobrazovací vrstvy.

5.1.4 Moduly **constants**, **requests**, **roles**, **errors**

V těchto modulech jsou nadefinovány různé konstanty. V modulu `constants` jsou nadefinovány veškeré konstanty použité napříč aplikací, kromě id požadavků, rolí a chyb, viz dále. Modul `roles` obsahuje konstanty představující role. V modulu `requests` jsou nadefinovány id požadavků. V modulu `errors` jsou nadefinovány všechny chyby, které mohou nastat během komunikace se serverem. Moduly `requests` a `errors` využívá komunikační protokol aplikace, viz *Kapitola 5.3*.

5.1.5 Modul **tools**

Modul poskytující různé pomocné funkce.

5.1.6 Modul **userView**

Modul `userView` se stará o zobrazení GUI a následnou zobrazovací logiku pro nepřihlášeného uživatele. Modul nejdříve načítá šablonu `userTemplate` a kaskádové styly `userStylesheet`. Poté inicializuje všechny GUI prvky a zajišťuje aktualizaci GUI podle volaných funkcí kontrolérem `userController`. V modulu je použito Google Maps API pro zobrazení mapy a vykreslení cest na dané mapě. Dále je využit modul `chartWindow`, který je popsaný v *Kapitole 5.1.8,* sloužící pro vizualizaci dat z daného senzoru v grafech. Modul `userView` dále poskytuje kontroléru hodnoty navolené na

nastavovacích prvcích a možnost předat vlastní obsluhu pro tlačítka vykonávající akci, které vyžadují komunikaci se serverem.

5.1.7 Modul **adminView**

Tento modul má stejnou funkci, jako modul předchozí s tím rozdílem, že se jedná o zobrazovací modul pro administrátora. Načítá tedy šablonu `adminTemplate` a kaskádové styly `adminStylesheet`. V modulu je také použito Google Maps API sloužící pro zobrazení pozic jednotlivých senzorů a cest a pro jejich případnou modifikaci. Modul dále zobrazuje tabulku obsahující dostupná data s informací, zda jsou staženy či nikoliv. U nestažených dat je k dispozici tlačítko pro stažení daných dat, jehož obsluha je nadefinována v kontroléru `adminController`.

5.1.8 Modul **chartWindow**

Modul `chartWindow` má na starosti vykreslování grafů a jejich nastavení v GUI pro nepřihlášeného uživatele. Základem tohoto modulu je objekt `InfoWindow` z Google Maps API, což je objekt představující okno, které se vykresluje na mapě. Dané okno se vykresluje u senzoru, na kterém bylo provedeno kliknutí. V okně jsou poté k dispozici nastavovací prvky, které slouží pro nastavení, co se má pomocí grafu z Google Charts API zobrazit.

5.2 Serverová část

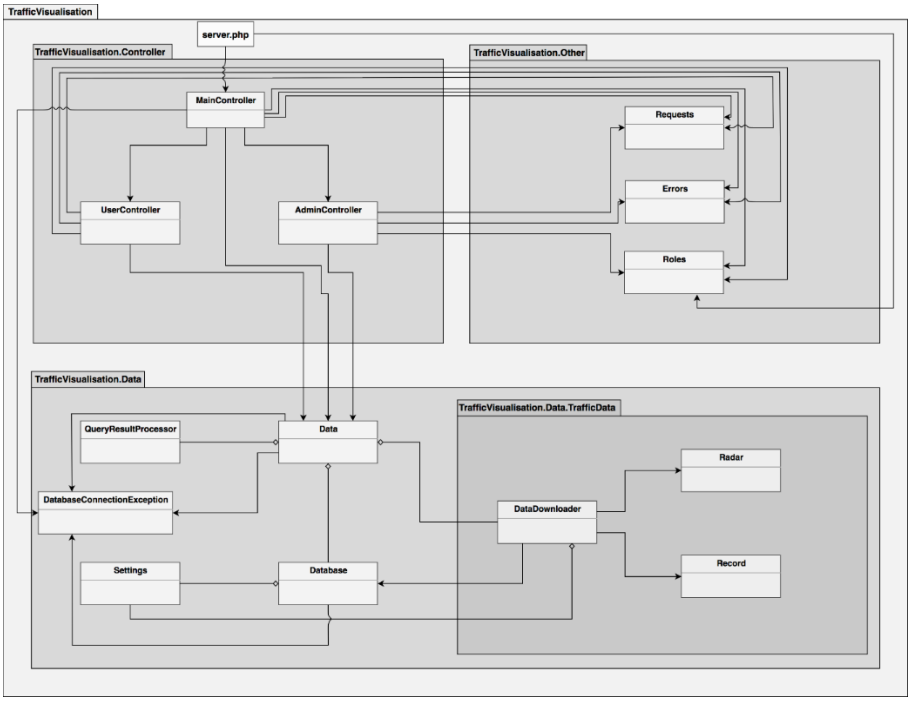
Serverová část je implementována v jazyce PHP. Vše je implementováno objektově, až na jedinou výjimku – skript `server.php`, což je skript, kterému klientské kontroléry předávají požadavky, viz *Kapitola 5.2.1*. Kód se dále dělí do tří základních jmenných prostorů – `Controller`, `Data` a `Other`.

Ve jmenném prostoru `Controller` je implementována serverová část kontroléru, která se mimo jiné také stará o přihlašování a odhlašování uživatele. V aplikaci není implementována registrace. V databázi je uložen pouze hash hesla vzniklý pomocí PHP funkce `password_hash()` s výchozím algoritmem, kterým je v současné době algoritmus Blowfish.

Ve jmenném prostoru `Data` je implementována datová vrstva. `Data` navíc obsahuje další jmenný prostor `TrafficData`, ve kterém jsou třídy, jejíž instance slouží pro stahování a zpracování dat z webových stránek obsahující silniční data.

Ve jmenném prostoru `Other` jsou ostatní třídy, jako například třída obsahující definice požadavků. Na *Obrázku 5.2* je znázorněna architektura serverové části aplikace.

V následujících podkapitolách je popsán skript `server.php` a následně třídy serverové části.



Obrázek 5.2: Architektura serverové části

5.2.1 Skript `server.php`

Tomuto skriptu se z klientské části posílají požadavky. Skript nejdříve zjistí, zda má uživatel přiřazenou nějakou roli. Pokud nemá, přiřadí mu výchozí roli, což je nepřihlášený uživatel. Role je uchovávána v `session` proměnné a uchovává se tedy i po obnově stránky. Poté předává požadavek instanci třídy `MainController`.

5.2.2 Třída `MainController`

Instance třídy `MainController` je kontrolér zpracovávající pouze jeden požadavek, a to požadavek na zjištění role. Všechny požadavky, kromě požadavku na zjištění role, obsahují roli uživatele, od kterého požadavek přišel, viz *Kapitola 5.3*. Tento kontrolér

z bezpečnostních důvodů kontroluje, zda se role v požadavku opravdu rovná roli uživatele uložené v `session` proměnné. Pokud kontrola proběhne úspěšně, předá se požadavek specifickým kontrolérům `AdminController` či `UserController`. Pokud kontrola proběhne neúspěšně, požadavek se nezpracuje a klientské části se zašle informace o chybě.

5.2.3 Třída `UserController`

Instance třídy `UserController` se starají o zpracování požadavků od nepřihlášeného uživatele. Většina požadavků se zpracovává zavoláním příslušné metody instance třídy `Data` s parametry poslanými klientskou částí. Návratová hodnota této funkce se poté odesílá zpět klientské části. Tento kontrolér se stará i o přihlašování do administrátorského prostředí.

5.2.4 Třída `AdminController`

Instance třídy `AdminController` zpracovávají požadavky od administrátora včetně odhlašování. Většina požadavků je opět zpracována zavoláním příslušné metody instance třídy `Data`.

5.2.5 Třídy `Requests, Roles, Errors`

Tyto tři třídy definují pouze statické konstanty, které jsou ekvivalentní těm v klientské části, viz *Kapitola 5.1.4*.

5.2.6 Třída `Data`

Instance třídy `Data` je vstupním bodem do datové vrstvy pro kontroléry. Instance poskytuje kontrolérům veškeré čtení, úpravu a vkládání dat do databáze a případně operace nad daty z webové stránky se silničními daty. Objekt obsahuje reference na další tři objekty, konkrétně instance tříd `Database`, `QueryResultProcessor` a `DataDownloader`, které jsou popsány v dalších kapitolách. Většina metod funguje následujícím způsobem – zavolá se příslušná metoda instance třídy `Database`, která navrátí výsledek dotazu nad databází. Tento výsledek se předá příslušné metodě instance třídy `QueryResultProcessor`, která výsledek dotazu zpracuje do formátu

vhodného pro odeslání klientské části. Zpracovaný výsledek dotazu se poté vrací kontroléru. Vyžaduje-li operace i práci s aktuálními silničními daty, je do procesu zakomponován i instance třídy `DataDownloader`, viz *Kapitola 5.2.10*.

5.2.7 Třída `Database`

Tato třída definuje objekt, který se stará o komunikaci přímo s databází. Objekt nejdříve navazuje spojení s databází přes `PDO` (PHP Data Objects), což je rozhraní pro pohodlnější práci s SQL databází.

Poté v jednotlivých metodách vytváří podle předaných parametrů příslušný dotaz, který se nad databází vykoná a jehož výsledek se vrací. Vykonávání dotazů nad databází se opět provádí přes `PDO`.

5.2.8 Třída `QueryResultProccessor`

Jak již bylo řečeno v *Kapitole 5.2.6*, instance třídy `QueryResultProcessor` slouží ke zpracování výsledků dotazů nad databází do formátu vhodného pro klientskou část.

5.2.9 Třída `Settings`

Třída Settings je implementována návrhovým vzorem Singleton neboli Jedináček, což umožňuje zajištění globálního přístupu k jedné instanci třídy. Tato instance obsahuje nastavení aplikace. Jednotlivým hodnotám nastavení je přiřazena buď hodnota specifikovaná v konfiguračním souboru, nebo hodnota výchozí. Výchozí hodnoty jsou definovány právě ve třídě `Settings`.

5.2.10 Třída `DataDownloader`

Instance třídy `DataLoader` poskytuje operace související s daty z webové stránky obsahující silniční data. Objekt poskytuje dvě metody – pro zjištění, jaká data jsou k dispozici a pro stažení dat z konkrétního data.

Zjištění, jaká data jsou k dispozici, probíhá následovně – stáhne se obsah HTML stránky, na které se data zveřejňují a regulárním výrazem, který je definován v nastavení aplikace, se naleznou veškerá dostupná data.

Metoda pro stažení dat z konkrétního data nejdříve konvertuje datum na název archivu obsahující daná data. Tento archiv se poté z webové stránky stáhne do složky s unikátním názvem a rozbalí. Složka musí mít unikátní název, jelikož se může stahovat více dat najednou a na konci procesu se složka s danými daty smaže. Unikátní název složky tedy zabraňuje, aby se smazala data, která dosud nebyla zpracována. Po rozbalení archivu se nejdříve zpracovává soubor se senzory. Kontroluje se, zda v souboru nepřibyl nový senzor. Pokud ano, daný senzor se přidá do databáze. Při tomto procesu se využívají instance třídy `Radar`. Dále se zpracovává soubor s danými daty. Při zpracování, jak již bylo řečeno, probíhá agregace a jsou využity instance třídy `Record`, které představují jeden agregovaný záznam.

5.2.11 Třída `Radar`

Instance třídy Radar představují právě jeden senzor. Třída je implementována návrhovým vzorem Messenger neboli Přepravka, což je pouze objekt obsahující několik atributů.

5.2.12 Třída `Record`

Instance této třídy představují jeden agregovaný záznam a jsou použity při zpracovávání nově stažených dat.

5.3 Komunikační protokol

Klientská a serverová část spolu komunikují pomocí AJAXu ve formátu JSON. Komunikace probíhá následujícím způsobem – klient odešle požadavek a server na něj odpoví výsledkem operace a případně daty.

5.3.1 Zaslání požadavku

Klient odesílá požadavky v následujícím formátu:

```
{
    "role": ROLE,
    "request": TYP_POZADAVKU,
    "data": DATA
}
```

Pole `role` obsahuje roli uživatele, od kterého požadavek přichází. Pole `request` obsahuje typ požadavku. Posledním polem je pole `data` obsahující data specifikující požadavek. Posílá-li se například požadavek na stažení a zpracování nových silničních dat, pole `data` obsahuje datum pořízení daných dat. Formát pole `data` je pevně spjatý s typem požadavku a není součástí každého požadavku.

5.3.2 Opověď na požadavek

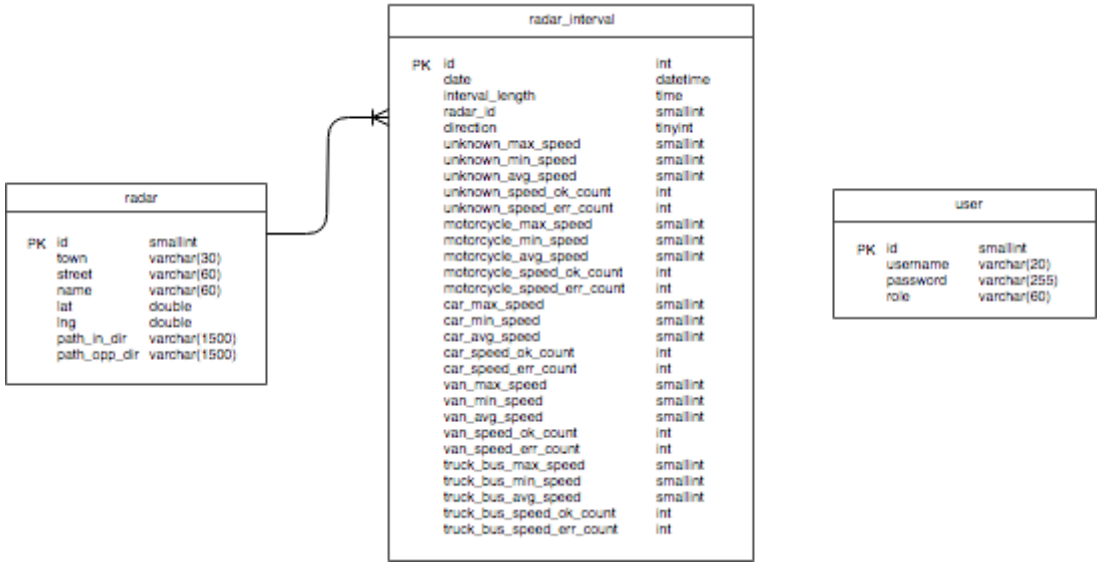
Server odesílá klientům odpověď v následujícím formátu:

```
{
   "error": TYP_CHYBY,
   "data": DATA
}
```

Pole `error` obsahuje kód případné chyby nebo kód, že byl požadavek zpracován bez chyby. Pole `data` obsahuje data, která byla vyžádána klientem. Pole `data` opět není součástí každého požadavku.

5.4 Popis databáze

Databáze se skládá celkem ze tří tabulek. Jedná se o tabulky `radar`, `radar_interval` a `user`. Na *Obrázku 5.3* je znázorněn ERA diagram tabulek databáze. Jednotlivé tabulky jsou poté popsány v následujících podkapitolách.



Obrázek 5.3: ERA diagram tabulek databáze

5.4.1 Tabulka `radar`

Záznamy v tabulce `radar` představují jednotlivé senzory. Tabulka obsahuje všechny složky souboru se senzory, viz *Kapitola 4.4.2*, kromě složky `IdArea`, která není pro aplikaci relevantní. Dále jsou součástí tabulky sloupce představující polohu senzoru a souřadnice cest.

5.4.2 Tabulka `radar_interval`

Tabulka `radar_interval` obsahuje všechny agregované záznamy. Tabulka se skládá z unikátního, automaticky generovaného id agregovaného záznamu a z dalších 29 sloupců, které jsou popsány v *Kapitole 4.5.2*.

5.4.3 Tabulka `user`

Záznamy v tabulce `user` představují uživatele. Tabulka se skládá celkem ze čtyř sloupců – unikátní id uživatele, uživatelské jméno, hash hesla a role uživatele.

6 Výsledky a testy

6.1 Porovnání agregovaného a neagregovaného způsobu uložení dat

V době návrhu aplikace bylo provedeno porovnání výkonnosti mezi agregovaným a neagregovaným způsobem uložení dat a zároveň testy správnosti provedení agregace. Porovnání probíhalo z důvodu rozhodnutí, zda využít agregaci dat či nikoliv. Testování probíhalo na počítači s následujícími parametry:

- Windows 10 Home, 64bit
- Intel Core i5-7300HQ, 2.50GHz
- 8GB RAM

Pro agregovaný způsob byla vytvořena tabulka popsaná v *Kapitole 4.4.2*. Pro neagregovaný způsob byla vytvořena tabulka s následujícími sloupci:

- `id` – unikátní id záznamu
- `radar_id` – id radaru, který záznam pořídil
- `date` – datum a čas pořízení záznamu
- `direction` – směr, ve kterém byl záznam pořízen
- `speed` – rychlost zaznamenaného vozidla
- `vehicle_type` – typ zaznamenaného vozidla

Porovnání výkonnosti a testy správnosti agregace probíhaly nad daty naměřenými za měsíc březen. Data byla již stáhnuta a extrahována z archívu, aby výsledky nebyly ovlivněny síťovým provozem. Všechny operace proběhly celkem desetkrát a střídal se agregovaný způsob s neagregovaným. V tabulkách jsou vždy uvedeny aritmetické průměry z naměřených hodnot.

Nejdříve bylo porovnáno, za jakou dobu se data zpracují a vloží do databáze. Výsledky jsou uvedeny v *Tabulce 6.1*. Z výsledků lze vidět, že agregovaný způsob je náročnější, co se týče zpracování a vkládání dat. Neagregovaný způsob je méně náročný, jelikož se při něm neprovádí téměř žádné zpracování dat (pouze výběr sloupců), na rozdíl od agregovaného způsobu.

Tabulka 6.1: Průměrný počet vteřin potřebných pro zpracování a vložení dat do databáze

Způsob uložení dat	Doba zpracování a uložení dat
Agregovaný způsob	516 s
Neagregovaný způsob	368 s

Dále byly nad tabulkami provedeny tři dotazy pro porovnání časové náročnosti získání výsledku a pro ověření správnosti provedení agregace. Následuje seznam popisů provedených dotazů:

- *První dotaz* – Počet všech zaznamenaných vozidel. Výsledky jsou uvedeny v *Tabulce 6.2*.
    - *Dotaz pro nad tabulkou obsahující agregovaná data*

```
SELECT
        (SUM(unknown_speed_ok_count)+  SUM(unknown_speed_err_count) +
        SUM(motorcycle_speed_ok_count)+ SUM(motorcycle_speed_err_count) +
        SUM(car_speed_ok_count) + SUM(car_speed_err_count) +
        SUM(van_speed_ok_count) + SUM(van_speed_err_count) +
        SUM(truck_bus_speed_ok_count) + SUM(truck_bus_speed_err_count)) as total
FROM
        radar_interval
```

    - *Dotaz pro nad tabulkou obsahující neagregovaná data*

```
SELECT
        COUNT(*) as total
FROM
        zaznamy
```

Tabulka 6.2: Výsledky a doba provedení prvního dotazu

Způsob uložení dat	Doba provedení dotazu	Počet vozidel
Agregovaný způsob	0,6479180 s	33 986 784
Neagregovaný způsob	6,7464612 s	33 986 784

- *Druhý dotaz* – Počet zaznamenaných osobních vozidel na radaru s id 51 a jejich průměrná rychlost ve směru k radaru. Výsledky jsou uvedeny v *Tabulce 6.3*.
    - *Dotaz pro nad tabulkou obsahující agregovaná data*

```
SELECT
        (SUM(car_speed_ok_count) + SUM(car_speed_err_count)) as total,
        AVG(car_avg_speed) as avg_speed
FROM
        radar_interval
WHERE
        radar_id='51' AND
        direction='1'
```

- *Dotaz pro nad tabulkou obsahující neagregovaná data*

```
SELECT
	COUNT(*) as total,
	AVG(speed) as avg_speed
FROM
	zaznamy
WHERE
	radar_id='51' AND
	vehicle_type='2' AND
	direction='1'
```

Tabulka 6.3: Výsledky a doba provedení druhého dotazu

Způsob uložení dat	Doba provedení dotazu	Počet vozidel	Průměrná rychlost
Agregovaný způsob	0,2918081 s	414 912	47,29 km/h
Neagregovaný způsob	6,5930130 s	414 912	46,63 km/h

- *Třetí dotaz* – Průměrné rychlosti osobních vozidel za jednotlivé dny v týdnu (pondělí – neděle) zaznamenaných mezi 8. – 22. hodinou na radaru s id 51 ve směru k radaru. Doby provedení dotazu jsou uvedeny v *Tabulce 6.4* a výsledky v *Tabulce 6.5*.
  - *Dotaz pro nad tabulkou obsahující agregovaná data*

```
SELECT
	weekday(date) AS weekday,
	AVG(car_avg_speed) AS avg_speed
FROM
	radar_interval
WHERE
	time(date) >= '08:00:00' AND time(date) < '22:00:00' AND
	radar_id='51' AND
	direction='1'
GROUP BY
	weekday
```

  - *Dotaz pro nad tabulkou obsahující neagregovaná data*

```
SELECT
	weekday(date) AS weekday,
	AVG(speed) AS avg_speed
FROM
	zaznamy
WHERE
	time(date) >= '08:00:00' AND time(date) < '22:00:00' AND
	radar_id='51' AND
	vehicle_type='2' AND
	direction='1'
GROUP BY
	weekday
```

Tabulka 6.4: Doby provedení třetího dotazu

Způsob uložení dat	Doba provedení dotazu
Agregovaný způsob	0,3711109 s

Neagregovaný způsob	4,4564619 s

Tabulka 6.5: Výsledky třetího dotazu

Den v týdnu	**Průměrná rychlost získaná dotazem nad tabulkou obsahující agregovaná data**	**Průměrná rychlost získaná dotazem nad tabulkou obsahující neagregovaná data**
Pondělí	45,42 km/h	45,54 km/h
Úterý	45,45 km/h	45,21 km/h
Středa	45,43 km/h	45,53 km/h
Čtvrtek	45,63 km/h	45,74 km/h
Pátek	46,00 km/h	46,05 km/h
Sobota	47,23 km/h	47,69 km/h
Neděle	48,38 km/h	48,67 km/h

6.2 Jednotkové testování

Aplikace byla otestována jednotkovými (*unit*) testy, k čemuž byl zvolen testovací framework PHPUnit[1]. Jednotkovými testy byla pokryta serverová část kontroléru, kde se především testovalo, zda jednotlivé kontroléry adekvátně reagují na příchozí požadavky. Dále byla pokryta část datové vrstvy, konkrétně třída `QueryResultProcessor`, u které se testovalo, zda správně formátuje výsledek dotazů do podoby vhodné pro klientskou část. Testy jsou dostupné na přiloženém CD, viz *Příloha D*.

6.3 Uživatelské testování použitelnosti

Pro uživatelské testování použitelnosti byl vytvořen scénář, podle kterého postupovali vybraní testeři. Scénář se nachází v *Příloze B*. Všichni testeři testovali aplikaci se stejnou databází. Skript pro vytvoření této databáze se nachází na přiloženém CD, viz *Příloha D*. Po splnění všech bodů scénáře bylo testerům umožněno volné používání aplikace.

[1] https://phpunit.de/

Bylo zvoleno celkem pět testerů. Požadavek na každého testera byl, aby měl alespoň základní znalosti v oblasti používání počítače. Čtyři testeři aplikaci testovali na počítači s následujícími parametry:

- OS: Windows 10 Home, 64bit (+ virtuální operační systém Ubuntu 18.04)
- Procesor: Intel Core i5-7300HQ, 2.50GHz
- RAM: 8GB

Poslední tester testoval aplikaci na počítači s následujícími parametry:

- OS: Mac OS X
- Procesor: Intel Core i5, 2,6 GHz
- RAM: 8 GB

Testerům bylo stručně vysvětleno, k čemu aplikace slouží. Aby se zjistilo, zda je používání aplikace intuitivní, nebylo testerům vysvětleno používání aplikace. Testeři aplikaci testovali v celkem pěti moderních webových prohlížečích. Každému testerovi byl přidělen jeden prohlížeč, ve kterém aplikaci testoval. Byl-li testerovi přidělen prohlížeč, který se v seznamu níže nachází u dvou operačních systémů, aplikaci otestoval v obou operačních systémech v daném prohlížeči. Testování bylo provedeno v následujících operačních systémech a webových prohlížečích:

- Windows 10 Home
  - Google Chrome, verze 67.0.3396.87
  - Mozilla Firefox, verze 60.0.2
  - Opera, verze 53.0.2907.106
  - Microsoft Edge, verze 42.17134.1.0
- Ubuntu 18.04
  - Google Chrome, verze 67.0.3396.87
  - Mozilla Firefox, verze 60.0.2
  - Opera, verze 53.0.2907.106
- Mac OS X
  - Safari, verze 11.1.1

Až na několik nalezených chyb a připomínek neměl žádný tester výraznější problém při plnění jakéhokoliv bodu ze scénáře, ani při následném volném používání. Nalezené chyby a vznesené připomínky byly následující:

- V prohlížečích Mozilla Firefox, Microsoft Edge a Safari se v legendě špatně zobrazuje barevná škála.
- Bylo by vhodné přidat možnost skrytí radarů a cest na mapě.
- V administrátorském prostředí jsem měl problém nalézt konkrétní radar. Bylo by vhodné radary seřadit podle abecedy.
- V prohlížečích Mozilla Firefox a Safari je v administrátorském prostředí špatně zarovnáno tlačítko „Potvrdit změny radaru“.
- Žádným způsobem se mi nepodařilo odstranit pozici a cesty radarů.

Nalezená chyba byla opravena a navrhnuté dodatečné funkcionality byly v aplikaci implementovány. Bylo tedy zajištěno, že aplikace funguje bez problémů na počítači v populárních webových prohlížečích.

Dále bylo mnou provedeno testování v mobilním webovém prohlížeči Google Chrome verze 67.0.3396.87 na mobilním zařízení LG G4 s operačním systémem Android 6.0. Testování probíhalo podle stejného scénáře, který byl použit u testování na počítači. Všechny kroky jsem úspěšně splnil, ale používání aplikace nebylo tak pohodlné, jako na počítači. Měl jsem problémy s kliknutím na tlačítka, pokud bylo zařízení ve vertikální poloze. Dále občas mizel kontrolní panel v administrátorském prostředí a stránka musela být obnovena, aby se panel opět objevil. Kritické chyby, jako právě mizení kontrolního panelu v administrátorském rozhraní, byly opraveny a aplikace je tedy použitelná i na mobilních zařízeních, i když ne v ideálním stavu. Optimalizace pro mobilní zařízení by tedy mohla být jedním z případných rozšíření aplikace.

7 Závěr

Cílem bakalářské práce bylo vytvořit systém pro vizualizaci dat o průjezdu vozidel z kamerového a radarového systému Plzeňského kraje. Systém je vytvořen a na počítačích je plně funkční.

Systém byl pojat jako webová aplikace umožňující běžnému uživateli vizualizovat data ze všech dostupných senzorů souhrnně na mapě. Dále lze vizualizovat data z konkrétního senzoru v grafech. Uživateli je umožněno provádět nad daty filtraci a může si tedy zvolit, jaká data budou ve vizualizaci zahrnuta.

Dále bylo navrženo a implementováno administrátorské rozhraní a přihlašovací systém. V administrátorském rozhraní je možné nastavovat určité parametry senzorů. Mezi nastavitelné parametry senzorů patří název daného senzoru, pozice senzoru na mapě a souřadnice cest směrem a v protisměru k radaru. Dále se v administrátorském prostředí zobrazují všechna dostupná denní data. Ty administrátor může nechat dát stáhnout, čímž spustí proces stahování, zpracování a následného uložení dat do databáze aplikace. Data se vzhledem k naměřeným výsledkům ukládají v agregované formě.

Aplikace byla otestována s využitím jednotkových testů a uživatelského testování použitelnosti. Z uživatelského testování použitelnosti vyplývá, že je aplikace bez problému použitelná v populárních moderních webových prohlížečích na počítačích. Poté bylo zjištěno, že aplikace není připravena pro používání na mobilních zařízeních. Optimalizace aplikace pro mobilní zařízení by tedy mohlo být jedním z rozšíření aplikace.

Přehled zkratek

HTML - HyperText Markup Language

SVG - Scalable Vector Graphics

MIT - Massachusetts Institute of Technology

API - Application Programming Interface

JSON - JavaScript Object Notation

CSS - Cascading Style Sheets

AJAX - Asynchronous JavaScript and XML

XML - Extensible Markup Language

CDN - Content Delivery Network

GUI - Graphical User Interface

SQL - Structured Query Language

PAM - Prvky Aktivního Monitoringu

CSV - Comma-Separated Values

PDO - PHP Data Objects

Literatura

[1] FEW, S. *„Eenie, Meenie, Minie, Moe: Selecting the Right Graph for Your Message“* [online]. Perceptual Edge. [cit. 2018/04/04] Dostupné z: http://www.perceptualedge.com/articles/ie/the_right_graph.pdf

[2] KOHOUT, J. *„Přednášky KIV/UPG“*. ZČU, Plzeň, 2017

[3] PANETH, N; VINTEN-JOHANSEN, P; BRODY, H; RIP, M. *„A rivalry of foulness: official and unofficial investigations of the London cholera epidemic of 1854“* [online]. American Journal of Public Health, 1998. [cit. 2017/11/13]. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1508470/

[4] BRADSHAW, P. *„When to use maps in data visualisation: a great big guide“* [online]. Online Journalism Blog, 2015. [cit. 2017/11/13]. Dostupné z: https://onlinejournalismblog.com/2015/08/24/when-to-use-maps-in-data-visualisation-a-great-big-guide/

[5] *„United States Patent and Trademark Office, registration #75263259“* [online]. USPTO, 1993. [cit. 2017/11/13]. Dostupné z: http://tsdr.uspto.gov/#caseNumber=75263259&caseType=SERIAL_NO&searchType=statusSearch

[6] *„Počet automobilů v ČR roste, loni jich bylo 485 na 1000 obyvatel“* [online]. Auto.cz, 2016. [cit. 2018/04/06]. Dostupné z: http://www.auto.cz/pocet-automobilu-v-cr-roste-loni-jich-bylo-485-na-1000-obyvatel-96735

[7] CHEN, W; F. GUO, F; WANG, F. Y. "*A Survey of Traffic Data Visualization*" [online]. IEEE, 2015. [cit. 2018/04/04]. Dostupné z: https://ieeexplore.ieee.org/document/7120975/

[8] *„Dopravní data“* [online]. Plzeňský kraj. [cit. 2018/04/05]. Dostupné z: http://doprava.plzensky-kraj.cz/site/page?view=od-traffic

[9] *„Using Google Charts“* [online]. Google. [cit. 2018/04/07]. Dostupné z: https://developers.google.com/chart/interactive/docs/

[10] *„Terms of Service“* [online]. Google. [cit. 2018/04/07]. Dostupné z: https://developers.google.com/terms/

[11] „*Chart.js*“ [online]. Chart.js. [cit. 2018/04/07]. Dostupné z: https://www.chartjs.org/

[12] SHARP, R. „*What is polyfill?*“ [online]. Remy Sharp’s blog, 2010. [cit. 2018/04/04]. Dostupné z: https://remysharp.com/2010/10/08/what-is-a-polyfill

[13] „*The MIT License*“ [online]. Open Source Initiative. [cit. 2018/04/04]. Dostupné z: https://opensource.org/licenses/MIT

[14] „*Kartografická zobrazení podle konstrukční osy*“ [online]. Mendelova univerzita v Brně. [cit. 2017/11/20], Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=59996

[15] „*Google Maps Projection*“ [online]. Setcompas. [cit. 2017/11/20]. Dostupné z: https://setcompass.com/GoogleMapsProjection.htm

[16] „*Splippy map*“ [online]. OpenStreetMap. [cit. 2017/11/20]. Dostupné z: https://wiki.openstreetmap.org/wiki/Slippy_Map

[17] „*Pricing details*“ [online]. Google. [cit. 2017/11/20]. Dostupné z: https://developers.google.com/maps/pricing-and-plans/#details

[18] „*Mapy API*“ [online]. Mapy.cz. [cit. 2017/11/27]. Dostupné z: https://api.mapy.cz/

[19] „*Usage of server-side programming languages for websites*“ [online]. W3Techs. [cit. 2018/04/20]. Dostupné z: https://w3techs.com/technologies/overview/programming_language/all

[20] KHOO M.; ROZAKLIS L.; HALL C.; KUSUNOKI D.; REGRIG M. „*Heat Map Visualizations of Seating Patterns in an Academic Library*“ [online]. Drexel University; The City University of New York. [cit. 2017/11/23]. Dostupné z: https://www.ideals.illinois.edu/bitstream/handle/2142/47285/274_ready.pdf

[21] „*Google Maps Documentation*“ [online]. Google. [cit. 2017/11/23]. Dostupné z: https://developers.google.com/maps/documentation/

[22] KISS, J. „*Secrets of a nimble giant*“ [online]. The Guardian, 2009. [cit. 2017/11/23]. Dostupné z:

https://www.theguardian.com/technology/2009/jun/17/google-sergey-brin

[23] *„O firmě“* [online]. Seznam.cz. [cit. 2017/11/27]. Dostupné z: https://onas.seznam.cz/cz/o-firme/historie-firmy/

[24] VÁCLAVÍK, L. *„Seznam má podrobnější mapu Evropy. Využil podklady z OpenStreetMap“* [online]. CNews, 2014. [cit. 2017/11/27]. Dostupné z: http://www.cnews.cz/seznam-ma-podrobnejsi-mapu-evropy-vyuzil-podklady-z-openstreetmap/

[25] *„HTML 5.3“* [online]. W3C. [cit. 2018/04/29]. Dostupné z: https://w3c.github.io/html/introduction.html

[26] *„What is JavaScript?“* [online]. Mozilla Foundation. [cit. 2018/04/29]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[27] *„AJAX“* [online]. Mozilla Foundation. [cit. 2018/04/29]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

[28] *„What is PHP?“* [online]. The PHP Group. [cit. 2018/04/29]. Dostupné z: http://php.net/manual/en/intro-whatis.php

[29] *„What is MySQL?“* [online]. Oracle Corporation. [cit. 2018/04/29]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Příloha A - Formáty zpracovávaných souborů

A.1 Ukázka souboru se senzory

```
"Česká Kubice, směr od Německa"|"Česká Kubice"|"Česká Kubice"|"KP055"|"5"
"Česká Kubice, směr od Babylonu"|"Česká Kubice"|"Česká Kubice"|"KP056"|"5"
"Holýšov, směr od Plzně"|"Holýšov"|"Jiráskova třída"|"KP051"|"7"
"Horšovský Týn, směr od Boru"|"Horšovský Týn"|"Masarykova"|"KP066"|"8"
"Horšovský Týn, směr od Stříbra"|"Horšovský Týn"|"Gorkého"|"KP067"|"8"
"Horšovský Týn, směr od Bělé"|"Horšovský Týn"|"Nová Ves"|"KP068"|"8"
"Klenčí pod Čerchovem, směr od Německa"|"Klenčí pod Čerchovem"|"Klenčí pod
Čerchovem"|"KP061"|"10"
"Klenčí pod Čerchovem, směr od Domažlic"|"Klenčí pod Čerchovem"|"Klenčí pod
Čerchovem"|"KP062"|"10"
"Libkov, směr od Klatov"|"Libkov"|"Libkov"|"KP069"|"11"
"Poběžovice, směr od Klenčí"|"Poběžovice"|"Slovanská"|"KP071"|"14"
"Staňkov, směr od H.Týna"|"Staňkov"|"Americká"|"KP053"|"16"
"Běšiny, směr od Klatov"|"Běšiny"|"Běšiny"|"KP072"|"18"
"Běšiny, směr od l/27"|"Běšiny"|"Běšiny"|"KP073"|"18"
"Červené Poříčí, směr od Plzně"|"Červené Poříčí"|"Červené Poříčí"|"KP059"|"19"
"Dlouhá Ves, směr od Sušice"|"Dlouhá Ves"|"Dlouhá Ves"|"KP058"|"20"
"Kolinec, směr od Sušice"|"Kolinec"|"Kolinec"|"KP057"|"25"
"Benešovice, směr od Stříbra"|"Benešovice"|"Benenšovice"|"KP060"|"32"
"Broumov, směr od Plané"|"Broumov"|"Broumov"|"KP065"|"34"
"Broumov, směr od Německa"|"Broumov"|"Broumov"|"KP064"|"34"
```

A.2 Ukázka souboru se záznamy

```
10058002|"2018-06-22 00:00:00.191"|1|1.00|100.00|67.00|0|2|0|""|2
10105201|"2018-06-22 00:00:00.579"|1|1.00|100.00|32.00|0|2|0|""|2
10106001|"2018-06-22 00:00:00.668"|1|1.00|100.00|33.00|0|2|0|""|2
10136001|"2018-06-22 00:00:00.641"|1|1.00|100.00|51.00|0|2|0|""|2
10196001|"2018-06-22 00:00:00.617"|1|1.00|100.00|47.00|0|2|0|""|2
10078001|"2018-06-22 00:00:01.832"|1|1.00|100.00|12.00|0|2|0|""|2
10081001|"2018-06-22 00:00:02.123"|1|1.00|100.00|47.00|0|2|0|""|2
10094101|"2018-06-22
00:00:02.325"|1|1.00|100.00|46.00|0|2|14|"0,46;1,48;2,48;3,49;4,49;5,48;6,47;7,46;"|2
10105001|"2018-06-22 00:00:02.613"|1|1.00|100.00|41.00|0|2|0|""|2
10136201|"2018-06-22 00:00:02.783"|1|1.00|100.00|52.00|0|2|0|""|2
10059001|"2018-06-22 00:00:03.332"|1|1.00|100.00|52.00|0|2|0|""|2
10063001|"2018-06-22 00:00:03.179"|1|1.00|100.00|38.00|0|2|0|""|2
10084102|"2018-06-22 00:00:03.948"|1|1.00|100.00|45.00|0|2|4|""|2
10094001|"2018-06-22 00:00:03.308"|1|1.00|100.00|48.00|0|2|0|""|2
10105201|"2018-06-22 00:00:03.309"|1|1.00|100.00|37.00|0|2|0|""|2
```

Příloha B - Testovací scénář

1. Na mapě vyhledejte radar ve Starém Plzenci (vpravo dole od Plzně) a rozklikněte jej

2. Nastavte datum a čas pro graf: od 1.4.2018 do 31.5.2018, zahrňte všechny dny v týdnu (Pondělí - Neděle) a celé dny (00:00 - 24:00)

3. Zobrazte si graf s počtem všech typů vozidel podle měsíců, s rozlišováním směru
   - Zobrazil se sloupcový graf s následujícími hodnotami?

Hodnota / Měsíc	Duben	Květen
Ve směru k radaru	371 067	415 441
V protisměru	266 185	289 138

4. Vyberte pouze pracovní dny (Pondělí - Pátek)

5. Zobrazte si graf s počtem osobních vozidel podle dnů, bez rozlišování směru
   - Zobrazil se sloupcový graf s následujícími hodnotami?

Hodnota / Den	Pondělí	Úterý	Středa	Čtvrtek	Pátek
Hodnota	208 282	183 251	224 492	227 379	159 147

6. Vyberte všechny dny (Pondělí – Neděle)

7. Zobrazte si graf s počtem všech typů vozidel podle roků, s rozlišováním směru.
   - Zobrazil se sloupcový graf s následujícími hodnotami?

Hodnota / Rok	2018
Ve směru k radaru	786 508
V protisměru	555 323

8. Nastavte čas na rozmezí 5:00 - 22:00

9. Zobrazte si graf s počtem všech typů vozidel podle času, bez rozlišování směru
   - Zobrazil se sloupcový graf s jednobarevnými sloupci, kde nejvíce naměřených vozidel je v intervalu od 15:30 a nejméně v intervalu od 5:00?

10. Zavřete okno s grafy a zobrazte si nové okno u radaru v "Nezvěstice" (vpravo dole od Starého Plzence).

11. Nastavte datum a čas podle bodu 2.

12. Zobrazte si průměrnou rychlost osobních vozidel podle času, s rozlišováním směru.

- Zobrazily se dva sloupcové grafy, pro které platí následující?
  - Ve směru k radaru je nejvyšší průměrná rychlost v intervalech začínajících ve 3:45 a 4:00 a nejnižší v intervalech začínajících v 15:15, 15:30 a 15:45
- V protisměru je nejvyšší průměrná rychlost v intervalu začínající ve 3:45 a nejnižší v intervalu začínající ve 4:00

13. Zobrazte si maximální rychlost všech typů vozidel podle dnů, bez rozlišování směru.

- Zobrazil se sloupcový graf se sedmi sloupci s hodnotami?

Hodnota / Den	**Pondělí**	**Úterý**	**Středa**	**Čtvrtek**	**Pátek**	**Sobota**	**Neděle**
Hodnota	151	157	156	141	160	127	138

14. Zavřete okno s grafy a zobrazte si nové okno u radaru v Přešticích (pod Plzní).

15. Datum a čas ponechte nastavený (podle bodu 2).

16. Zobrazte si typy vozidel s rozlišováním směru.

- Zobrazily se dva výsečové grafy s následujícími hodnotami?

Hodnota / typ vozidla	**Osobní vozidla**	**Dodávky**	**Kamiony a autobusy**
Ve směru k radaru	831 300 (94,8 %)	7 474 (0,8 %)	43 879 (4,5 %)
V protisměru	89 187 (91,7 %)	30 659 (3,1 %)	50 382 (5,1 %)

17. Zobrazte si typy vozidel bez rozlišování směru.

- Zobrazil se jeden výsečový graf s následujícími hodnotami?

Hodnota / typ vozidla	**Osobní vozidla**	**Dodávky**	**Kamiony a autobusy**
Hodnota	1 829 487 (93,3 %)	38 133 (1,9 %)	94 261 (4,8 %)

18. Zavřete okno a otevřete si nastavení cest (následující tři body se týkají zobrazení pomocí cest).

19. Nastavte datum a čas podle bodu 2.

20. Zahrňte všechny typy vozidel a zobrazte počty vozidel.
    - Jsou u legendy hodnoty od 6 372 až 1 342 271?
    - Je u radaru v Staňkově (vlevo od Přeštic) nejméně zaznamenaných vozidel (nejzelenější cesta)?
    - Je u radaru v Losiné (vpravo dole od Plzně) nejvíce zaznamenaných vozidel (nejčervenější cesta)?

21. Zahrňte pouze osobní vozidla a zobrazte si průměrnou rychlost.
    - Jsou u legendy hodnoty od 20 km/h až 80 km/h?
    - Je u radaru v "Staňkov" (vlevo od Přeštic) nejmenší průměrná rychlost (nejzelenější cesta)?
    - Je u radaru v "Kozolupy" (vlevo od Plzně) největší průměrná rychlost (nejčervenější cesta)?

22. Přihlaste se do administrátorského prostředí - uživatelské jméno i heslo je "admin".

23. Zobrazte si seznam radarů.

24. Proveďte u radaru "Přeštice" změnu názvu na libovolný jiný název.

25. Odhlaste se z administrátorského rozhraní a rozklikněte si radar v Přešticích (pod Plzní).
    - Změnil se název radaru na Vámi zvolený v předchozím kroku?

26. Nalezněte na mapě Červené poříčí (pod Přešticemi).
    - Je u lokace viditelný radar nebo vykreslené cesty? (Ani jedno by viditelné být nemělo).

27. Přejděte opět do administrátorského rozhraní a v seznamu radarů nalezněte radar s názvem "Červené poříčí, směr od Plzně" (měl by to být jediný radar bez nastavených souřadnic a cest) a nastavte souřadnice a obě cesty poblíž dané lokace (přesnost není důležitá)

28. Odhlaste se z administrátorského rozhraní.

29. Je již u "Červené poříčí" vykreslený radar a cesty, které jste nastavili?

30. Zkontrolujte, zda lze u grafů a cest nastavit maximální rozmezí dat od 1. 4. 2018 do 31. 5. 2018.

31. Přihlaste se do administrátorského rozhraní a rozklikněte záložku data a vyčkejte na případné načtení dat.

- Jsou v tabulce data od 1. 1. 2016 do včerejšího/předvčerejšího data?

32. Přihlaste se do administrátorského rozhraní a stáhněte data z 31. 3. 2018 a 1. 6. 2018. Vyčkejte, dokud se obě stahování nedokončí.

33. Odhlaste se z administrátorského rozhraní.

34. Zkontrolujte, zda lze u grafů a cest nastavit maximální rozmezí dat od 31. 3. 2018 do 1. 6. 2018.

35. Zobrazte libovolný graf či cesty z 31. 3. 2018 a poté z 1. 6. 2018.

- Zobrazila se data?

Příloha C - Uživatelská příručka

C.1 Instalace aplikace

V této kapitole je podrobně popsán návod, jak lokálně zprovoznit aplikaci s využitím volně dostupného softwarového balíku XAMPP v operačním systému Windows 10. V kapitole se předpokládá, že je XAMPP již nainstalovaný.

- Ve složce htdocs, která se nachází v kořenovém adresáři balíčku XAMPP, vytvořte novou složku s názvem trafficVisualisation (název může být i jiný, dále se ale bude předpokládat, že byl zvolen tento název).
- Do Vámi vytvořené složky v předchozím kroku vložte složku app nacházející se na přiloženém CD. Dále do složky vložte soubor usage_example.html, který se také nachází v přiloženém CD.
    - Místo souboru usage_example.html můžete použít vlastní HTML soubor. V hlavičce Vašeho HTML souboru naimportujte aplikaci následujícím HTML tagem:

```
<script src="app/src/trafficVisualisation.js"></script>
```

    - Následně vytvořte v těle HTML souboru obalový prvek `DIV`, s `id` například `vizualizace`. Poté už můžete inicializovat aplikaci následujícím voláním (proměnná gmapsApiKey představuje textový řetězec obsahující Google Maps API klíč)

```
transitVisualisation.main.init(
document.getElementById("vizualizace"), gmapsApiKey);
```

- Aby se předešlo problémům v následujících krocích, je nutné provést změny v nastavení. Otevřete soubor php.ini, který se nachází ve složce xampp/php (xampp představuje adresář, kde je XAMPP nainstalovaný) a v souboru změňte následující hodnoty
    - max_execution_time = 5000
    - max_input_time = 5000
    - memory_limit = 1000M
    - post_max_size = 750M

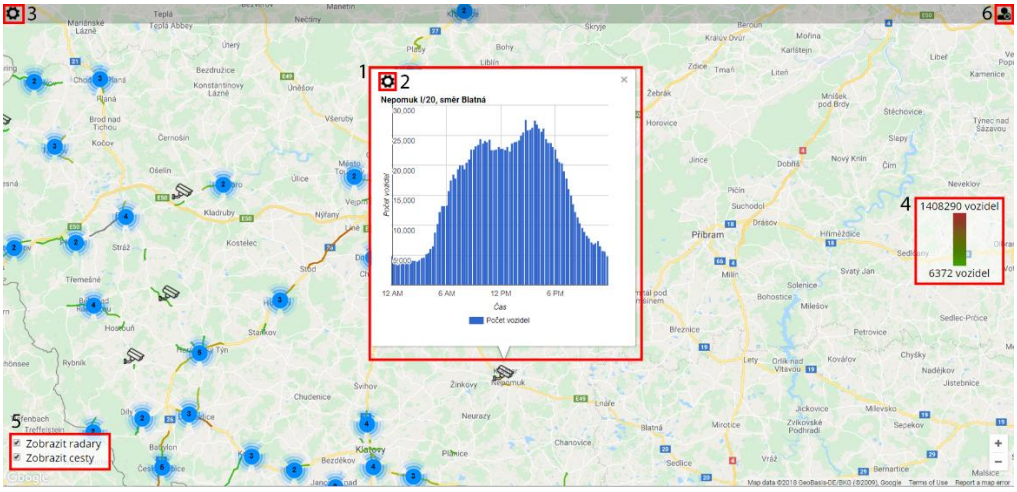
    - upload_max_filesize = 750M
- Zapněte XAMPP Control Panel a klikněte na start u Apache a MySQL.
- Ve webovém prohlížeči přejděte na adresu http://localhost/phpmyadmin/
- V horním menu klikněte na záložku „import“.
- Klikněte na tlačítko „Choose file“ a v dialogu přejděte do složky db_scripts nacházející se na přiloženém CD. Složka obsahuje několik skriptů pro vytvoření databáze a soubor README.txt, ve kterém jsou jednotlivé skripty popsány.
- Vyberte jeden ze skriptů, klikněte na tlačítko „Go“ a vyčkejte na dokončení vykonávání skriptu.
- V souboru config.cfg, který se nachází ve složce app, lze případně nastavit aplikaci. Nastavit lze například server a přihlašovací údaje k databázi, pokud aplikaci nepoužíváte lokálně pomocí balíku XAMPP.
    - Změna nastavení není nutná, pokud chcete aplikaci používat pouze lokálně pomocí balíku XAMPP.
- Nyní je vše připraveno a můžete aplikaci začít používat ve webovém prohlížeči na adrese http://localhost/trafficVisualisation/usage_example.html (v případě, že jste ve druhém kroku zvolil předpřipravený soubor usage_example.html)

C.2 Používání aplikace

Při prvotním spuštění aplikace lze aplikaci používat jako nepřihlášený uživatel. Pokud byl při instalaci zvolen skript, který vyplňuje tabulku uživatelů, lze se přihlásit do administrátorského prostředí (výchozí jméno i heslo je „admin“).

C.2.1 Nepřihlášený uživatel

Nepřihlášenému uživateli si může zobrazit data dvěma způsoby – pomocí cest a pomocí grafů. GUI pro nepřihlášeného uživatele lze vidět na *Obrázku C.1*, který je následně vysvětlen.



Obrázek C.1: GUI pro nepřihlášeného uživatele

Popis Obrázku C.1:

1 Okno zobrazené u právě vybraného radaru na mapě, ve kterém je název vybraného radaru a vizualizovaná data zaznamenaná daným radarem.

2 Tlačítko zobrazující nastavení grafů.

3 Tlačítko zobrazující nastavení vizualizace na mapě pomocí cest.

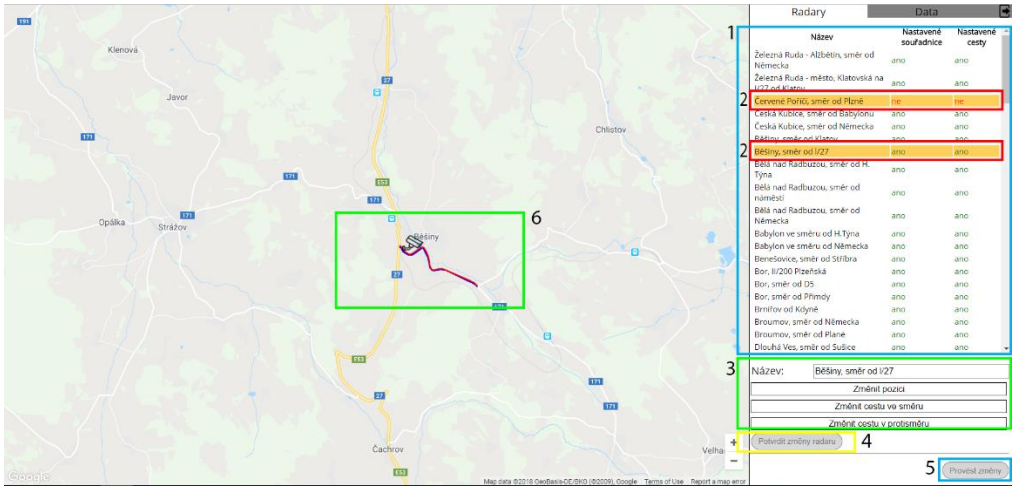
4 Legenda k vizualizaci na mapě.

5 Možnosti zobrazení či skrytí radarů a cest na mapě.

6 Tlačítko zobrazující dialog pro přihlášení do administrátorského prostředí.

C.2.2 Administrátor

Administrátor má k dispozici dvě funkce – spravování radarů a stahování nových dat. Spravování radarů lze vidět na *Obrázku C.2* a stahování dat na *Obrázku C.3*. Obrázky jsou následně opět vysvětleny.



Obrázek C.2: GUI pro administrátora – nastavení radarů

Popis *Obrázku C.2*:

7 Seznam všech radarů a stav nastavení jejich pozic a souřadnic cest.

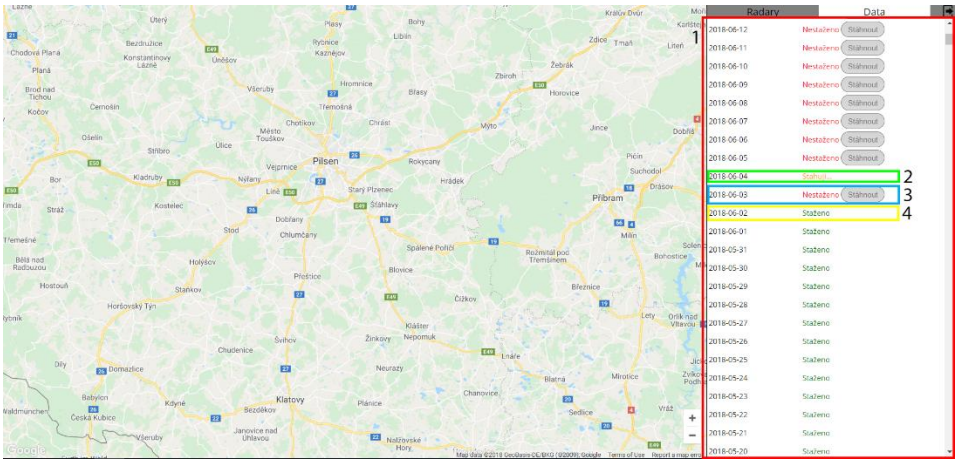
8 Radary, u kterých byly potvrzeny změny.

9 Panel, kde se provádí změny radaru.

10 Tlačítko, kterým se potvrdí změny radaru. Pokud se u radaru provedou cesty a nebudou potvrzeny (klikne-li se například na jiný radar), tak budou změny zahozeny.

11 Tlačítko, kterým se provedou potvrzené změny radaru.

12 Vyznačený současně vybraný radar, včetně nastavených cest.



Obrázek C.3: GUI pro administrátora – stahování dat

Popis *Obrázku C.3*:

1 Seznam všech dostupných denních dat.

2 Právě stahovaná data.

3 Nestažená data s tlačítkem pro stažení daných dat.

4 Již stažená data.

Příloha D - Obsah přiloženého CD

- **`app/`** – Obsahuje zdrojové kódy aplikace.
- **`app_test/`** – Obsahuje zdrojové kódy jednotkového testování.
- **`db_scripts/`** – Obsahuje skripty pro vytvoření databáze a soubor README.txt obsahující popis daných skriptů.
- **`docs/`** – Obsahuje tuto práci ve formátu PDF a zdrojový kód textu.
- **`usage_example.html`** – Ukázka použití aplikace.
- **`README.txt`** – Popis obsahu CD.