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

Diplomová práce

Vizualizace dat a mashup aplikace

Plzeň, 2013 Michal Smeták

Poděkování

Děkuji Ing. Pavlu Královi, Ph.D. vedoucímu této diplomové práce za jeho cenné rady a čas který mi věnoval.

Dále bych chtěl poděkovat rodičům a všem svým blízkým za jejich morální i finanční podporu při studiu, trpělivost a motivaci.

Prohlášení

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

V Plzni dne 9. května 2013 .....

Michal Smeták

Abstract

This diploma thesis is based on the requirements of Czech News Agency (ČTK). The first goal of this work is to analyze data structure of ČTK and existing software tools for data vizualization with a particular focus on the free available tools with posibilities of commercial use. Based on this analysis, the most promissing tools are selected. The second goal consists of the desing and implementation of two web applications for data vizualization. The third goal solves the problem of data clustering for data vizualization.

The first application is used for geographical data representation in several ways, the second application uses data representation on the timeline. These two applications allow to vizualize the ČTK data of the incidents and the documents. The third goal is to implement a modified clustering algorithm, which creates clusters respecting the geographical distance. This algorithm is used in the first application. Clustering of the data in the second application is a part of tool used for vizualization.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní době se naprostá většina dat uchovává v digitální podobě. Denně vznikají nová data (zpravodajské články, fotografie, videa, záznamy různých událostí apod.), která je třeba archivovat. Aby bylo možné stále narůstající množství dat efektivně spravovat, prohledávat či vizualizovat, přidávájí se k datům samotným takzvaná metadata. Metadata obsahují dodatečné informace o datech a usnadňují tak jejich správu.

Tato diplomová práce vyplynula z potřeb České tiskové kanceláře (ČTK). Cílem bylo na základě analýzy struktury metadat zpravodajských servisů ČTK a zdrojů dat dostuných na webu navrhnout a implementovat webové aplikace pro vizualizaci zvolených dat pomocí vhodných volně dostupných nástrojů s možností komerčního použití.

Výsledek této práce by měl poskytnou přehled volně dostupných nástrojů, zaměřených na geografickou vizualizaci dat a nástrojů pro vizualizaci dat formou časových řad. Dalším výstupem práce bude analýza možností vizualizace dat z jiných dostupných datových zdrojů (ČSÚ apod.). Aplikace implementované ve vybraných nástrojích, které budou splňovat požadavky zadavatele, budou zároveň obsahovat i návrhy řešení problému přehledného grafického zobrazení velkého počtu dat.

Zadavatel nepředpokládá přímé využití vytvořených webových aplikací. Vytvořené aplikace budou zadavateli sloužit pro inspiraci jak daná data zobrazovat, případně jako zdroj kódu při vytváření vlastních aplikací.

1.1 Přehled kapitol

V úvodní kapitole jsou představeny důvody vzniku této práce, cíle a předpokládané budoucí využití. Druhá kapitola se zabývá analýzou a popisem struktury metadat ČTK. Třetí kapitola obsahuje popis mapových serverů a běžně používaných webových služeb pro získání dat (mapových podkladů). Při používání mapových API [1] je vhodné být s touto problematikou seznámen. Následující kapitola se zabývá zdroji dat na webu. Z větší části se tato kapitola zabývá popisem volně dostupných mapových serverů, které nemají žádná funkční omezení. Kapitola 5 obsahuje analýzu volně dostupných nástrojů pro geografickou a časovou vizualizaci dat. V závěru kapitoly jsou čtenáři nabídnuty tabulky porovnávající vlastnosti popsaných nástrojů a je zde uvedeno zdůvodnění výběru nástrojů použitých pro realizaci webových aplikací. Následující kapitola se zabývá realizační částí této práce. Obsahuje návrh aplikace a popis jednotlivých částí. V kapitole 7 je popis testování použitého shlukovacího algoritmu, dosažené výsledky implementovaných aplikací a návrhy na jejich případná rozšíření. Závěrečná kapitola shrnuje výběr použitých nástrojů, dosažené výsledky implementovaný aplikací a časovou náročnost použitého shlukovacího algoritmu.

Přílohy obsahují strukturu přiloženého DVD, strukturu vytvořené aplikace a uživatelskou příručku.

[1] Application Programming Interface

Kapitola 2

Analýza metadat ČTK

Metadata ČTK se dají rozdělit na dva typy: *dokumenty* (texty, články, zpravodajství, apod.) a *události* (záznamy událostí, výročí, plánovaných akcí). Pro oba typy dat (*dokumenty i události*) je definován formát založený na standardu XML. Výhoda XML formátu je především v jeho přehlednosti, snadném zpracovávání a provádění transformací na straně uživatele, či možnosti přímého využití na webových stránkách. *Dokumenty* jsou poskytovány ve speciálně navržené XML struktuře, která vychází ze standardu *NewsML 1.2* [23] definovaném organizací IPTC (International Press Telecommunications Council). Standard umožňuje uchovávat k jednotlivým záznamům velké množství strukturovaných metadat, které uživateli nabízí možnost dokumenty dále zpracovávat, např. je kategorizovat či vizualizovat. Standard *NewsML* používá koncept slovníků. Slovníky mohou přesně určovat hodnoty, které se mohou v některých položkách formátu objevovat. Zároveň také popisují jejich význam. ČTK používá vlastní slovníky[1] i slovníky definované IPTC, které rozšiřuje o vlastní hodnoty. Definice přiřazení slovníků k elementům či atributům formátu *NewsML* určuje katalog[2][22].

Pro *události* má ČTK definovaný vlastní XML formát, který také používá slovníky. Je ale jinak strukturovaný.

Metadata pro *dokumenty* a *události* nejsou stejná. Některé elementy metadat obou typů mají sice stejný význam, nicméně jejich struktura se liší. Jednotlivé formáty metadat jsou proto popsány v následujících dvou podkapitolách.

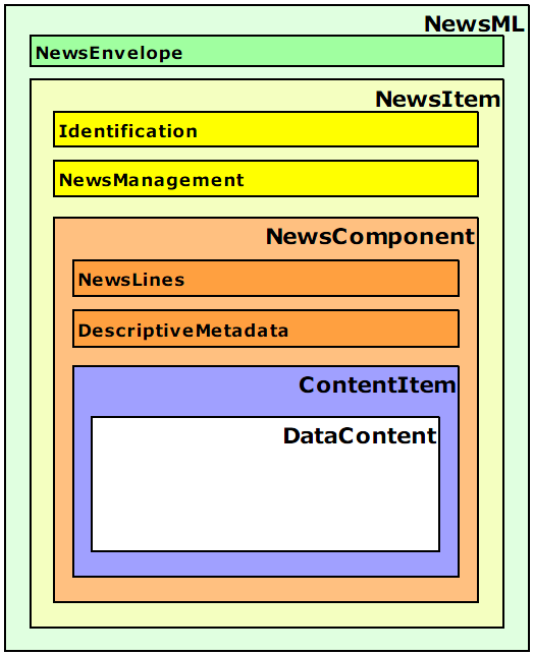
2.1 Dokumenty

Základní struktura formátu *dokumentů* je na obrázku 2.1. Formát *NewsML* je snadno rozšiřitelný použitím elementů *property*. Tabulka z rozšiřujícími vlastnostmi pro element *property* je dostupná v [22].

Každý *dokument* je uložen v samostaném elementu *NewsItem*. Uvnitř *NewsItem* jsou vnořeny elementy *Identification, NewsManagement a NewsComponent*.

[1] http://newsml.ctk.cz/topicset/

[2] http://newsml.ctk.cz/catalog/CTK_Catalog.xml



Obrázek 2.1: Základní struktura formátu dokumentů NewsML [22]

Identification

Element *Identification* obsahuje data sloužící k idetifikaci *dokumentu*. Ve své aplikaci jsem využil tyto elementy:

- *DocumentType* - typ *dokumentu* (text, video, fotografie, ...).
- *DateId* - datum vytvoření *dokumentu* ve formátu *YYYY-MM-DD*.
- *PublicIdentifier* - unikátní veřejný identifikátor *dokumentu*.

NewsManagement

Metadata elementu *NewsManagement* obsahují informace o *dokumentu*, např. typ, historii, číslo revize, status, vazby na ostatní *dokumenty* a pod.

NewsComponent

Element *NewsComponent* zapouzdřuje samotný obsah *dokumentu*. Jsou v něm vnořeny elementy *Metadata* a *ContentItem*.

Element ***DescriptiveMetadata*** v sobě obsahuje metadata vztahující se k *dokumentu* (např. autora, editory, kategorie do kterých *dokument* spadá, jeho nadpis, zdroj a další). Pro mne byly nejvýznamější z tyto elementy:

- *Headline* - nadpis.

- *Caption* - popisek.
- *Category* - kategorie do které *dokument* patří (může se vyskytovat v metadatech vícekrát).
- *Location* - popis geografického umístění pomocí vnořených elementů *Region, Country, City, Place a Position* (s atributy GPS souřadnic). Viz následující ukázka:

```
<Location>
  <Region>ce</Region>
  <Country>ČR</Country>
  <City>Praha</City>
  <Place>
      Valdštejnský palác, Valdštejnské nám. 4, Praha 1
  </Place>
  <Position latitude="50.09008" longitude="14.405426" source
     ="auto" />
</Location>
```

Element ***ContentItem*** popisuje samotný obsah *dokumentu*. Pokud se jedná o textový dokument, je text *dokumentu* uložen v elementu *BodyContent*. *BodyContent* může obsahovat i některé XHTML značky, jejich výčet je v tabulce 2.1. V případě binárních *dokumentů* (foto, video, audio) obsahuje element *ContentItem* atribut *Href*, který odkazuje na binární soubor. Vnořené elementy v *ContentItem* jsou:

- *MediaType* - typ *dokumentu* [3] (*Text, Photo, Audio, ...*).
- *Format* - formát binárního souboru (pokud se nejedná o *MediaType Text*).
- *Charasteristics* - charakteristika *dokumentu* (např. počet slov, odstavců apod. u textového *dokumentu*, nebo velikost, název souboru, rozměry atd. u binárního *dokumentu*).
- *BodyContent* - vlastní obsah textového *dokumentu*.

2.2 Události

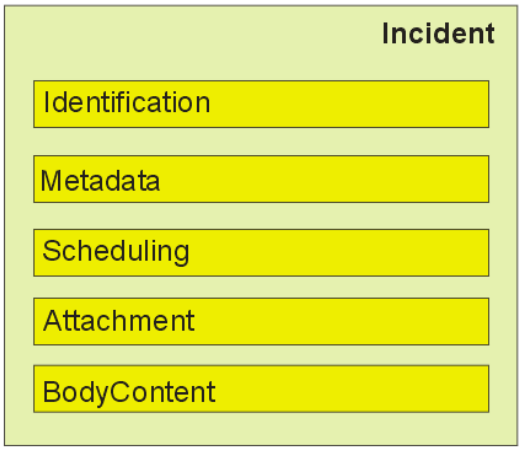
Soubory s *událostmi* obsahují záznamy událostí, výročí a plánovaných akcí. Součástí *událostí* mohou být i multimediální přílohy (foto, audio, video). Struktura XML formátu pro jejich uchovávání je popsána dále a zobrazena na obrázku 2.2.

Každá *událost* je umístěna v samostatném elementu *Incident*. Následuje popis základních elementů *události*:

[3] Možné hodnoty jsou definované na http://newsml.ctk.cz/topicset/topicset.ctkvalue-documenttype.xml

<p>	odstavec
<a>	hypertextový odkaz
<table>	tabulka
<tr>	řádek tabulky
<td>	buňka tabulky
<tbody>	tělo tabulky

Tabulka 2.1: Výčet přípustných XHTML značek v elementu *BodyContent*



Obrázek 2.2: Základní struktura formátu pro události

- *Identification* - stejný význam jako u *dokumentů* (identifikace *události*). Má podobnou strukturu, pouze obsahuje méně elementů.
- *Metadata* - stejný význam jako element *DescriptiveMetadata* u *dokumentů*, liší se pouze v některých elementech.
- *Scheduling* - obsahuje časové údaje o události. Element *DateStart* udává datum (případně i čas), kdy má daná událost nastat. Formát data je ve tvaru *YYYY-MM-DD hh:mm*.
- *Attachment* - multimediální příloha. Strukturou odpovídá elementu *ContentItem* pro binární data *dokumentů*, liší se pouze názvem. Obsahuje atribut *Href* odkazující na přiložený soubor a vnořené elementy *MediaType, Format a Characteristics*.
- *BodyContent* - textový obsah, totožný formát jako u *dokumentů*.

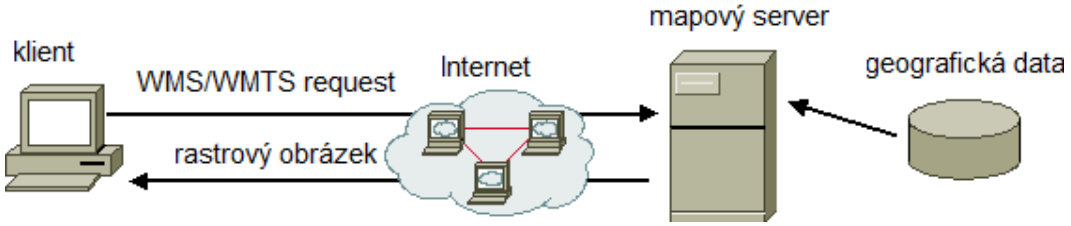
2.3 Shrnutí

Data ČTK jsem se po analýze metadat rozhodl primárně vizualizovat na základě časových údajů, geografických údajů a jejich kombinací. Časové údaje vhodné k vizualizaci formou časové osy byly obsažené ve všech exportovaných položkách. Geografické údaje - souřadnice GPS (Global Positioning System), vhodné pro vizualizaci dat na mapě, se v ČTK používají prozatím pouze v testovacím provozu. Proto v některých souborech zcela chyběly.

Kapitola 3

Mapové servery

Mapový server je klient-server aplikace, která je schopna na základě požadavku a dostupných geografických dat vygenerovat odpovídající mapové podklady, nejčastěji v rastrovém formátu (formáty PNG, TIFF, JPEG, GIF). Schéma komunikace je na obrázku 3.1. Geografická data potřebná pro vygenerování mapy jsou georeferencována (mají jednoznačně daný souřadnicový systém). Díky tomu lze kombinovat více druhů geografických dat během generování mapových podkladů. Mapový server může podporovat jednu či více webových mapových služeb, které jsou definovány standardy Open Geospatial Consorcia (OGC) [24].



Obrázek 3.1: Komunikace mezi klientem a mapovým serverem

3.1 Web Map Service

Web Map Service (WMS) je OGC standard, který poskytuje jednoduché HTTP rozhraní pro dotazování geografických mapových dat uložených v lokální nebo distribuované databázi. WMS dotaz definuje geografickou vrstvu a oblast zájmu. Odpovědí na WMS dotaz jsou rastrové obrázky – georeferencovatelné mapové podklady, které mohou být zobrazeny v klientské aplikaci. Rozhraní také umožňuje specifikovat, zda vrácené obrázky mají být průhledné, což umožňuje kombinovat data v distribuovaném systému složeném z více serverů [24].

3.1.1 GetMap request

GetMap request je jádro WMS protokolu. Jedná se o specifickou HTTP Key-Value-Pair (KVP) adresu, která vrátí obrázek, či obrázky dotazované mapy. Dotaz se skládá z URL WMS mapového serveru, verze protokolu, typu dotazu, rozměrů ohraničovacího rámečku (4 souřadnice: xMin, yMin, xMax, yMax – x odpovídá zeměpisné délce, y zeměpisné šířce), typu projekce, velikosti obrázku, typů vrstev které chceme zobrazit a formátu obrázku v kterém chceme zaslat odpověď.

Odpověď bude obrázek v definovaném formátu o daných rozměrech, z místa specifikovaném v ohraničujícím rámečku [12].

Ukázka URL pro GetMap:

```
<ServiceRoot>?Service=WMS&Version=1.1.0&Request=GetMap&BBox
    =-20,-40,60,40&SRS=EPSG:4326&Width=400&Height=800&Layers=
    Countries&Format=image/gif
```

3.1.2 GetCapabilities request

K tomu abychom byli schopni složit URL pro GetMap request, potřebujeme znát informace o mapových datech na serveru. Např. které vrstvy jsou na serveru dostupné, jaký souřadnicový systém používá a jaké typy obrázků a verze protokolu podporuje. Odpovědí na tento dotaz je strukturovaný XML soubor s informacemi o WMS serveru a geografických datech která poskytuje [12].

Ukázka URL pro GetCapabilities:

```
<ServiceRoot>?request=GetCapabilities&version=version
```

3.2 Web Map Tile Service

Web Map Tile Service (WMTS) je implementační OGC standard, který poskytuje reprezentaci digitální mapy na základě předem vygenerovaných obrázkových dlaždic uložených na serveru. Dlaždice jsou uloženy v adresářové struktuře podle typu mapy, vrstvy, úrovně zoomu, jejich pozice apod. Každému typu mapy, úrovni zoomu atd. bude odpovídat jiný adresář. Odtud je pak server poskytuje na vyžádání klientům. WMTS má k dispozici standardizovaný dokument *ServiceMetadata.xml*, který strukturu dat na serveru popisuje. Definuje dostupné dlaždice v každé vrstvě, v každém referenčním souřadném systému, v každém měřítku apod. Na základě informací v dokumentu *ServiceMetadata.xml* mohou klienti požádat o konkrétní dlaždice. WMTS

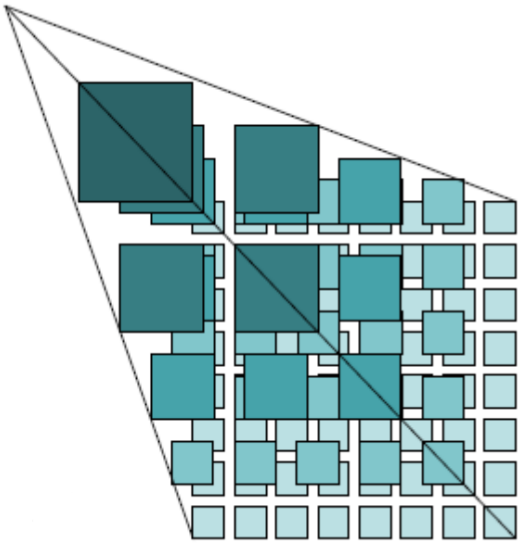
používají všechny moderní mapové aplikace, jejich výhodou oproti WMS je rychlost poskytování dlaždic a snížená zátěž serveru. Na HTTP požadavek se odpoví konkrétní dlaždicí (souborem na disku). U WMS se musí obrázky generovat dynamicky [1].

3.2.1 GetTile request

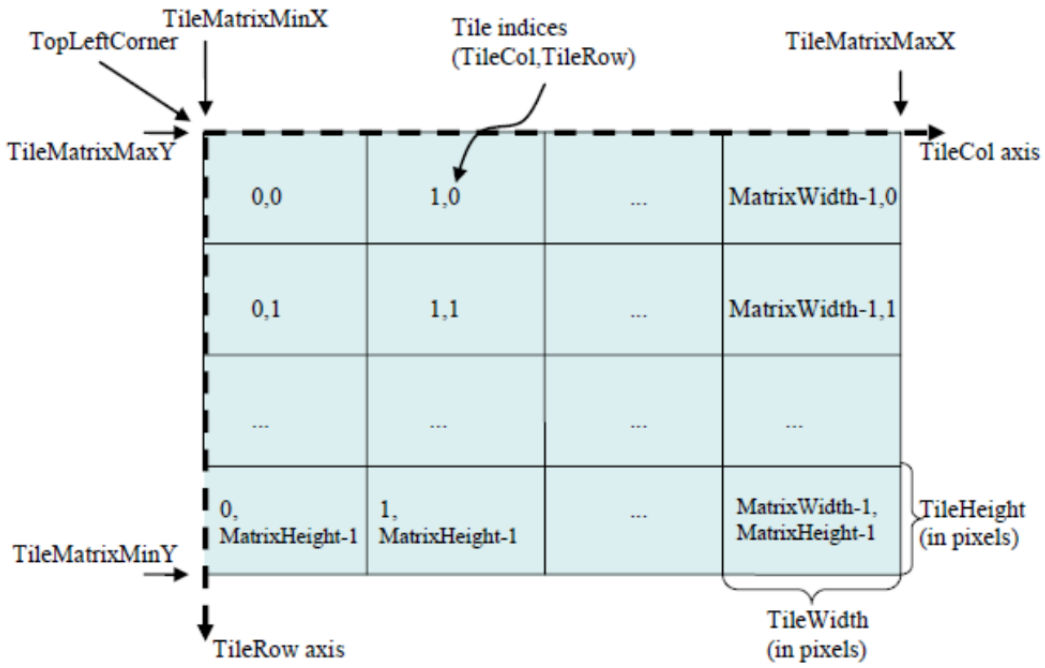
GetTile request je HTTP KVP dotaz na konkrétní dlaždici. Dotaz se skládá z URL WMTS mapového serveru, verze protokolu, typu dotazu, druhu požadované vrstvy, stylu, formátu dlaždic, typu projekce, úrovně zoomu (TileMatrix viz obrázek 3.2) a čísla řádku a sloupce v dlaždicovém prostoru (viz obrázek 3.3). Dlaždicový prostor tvoří jednu vrstvu v TileMatrix. Nultá vrstva na obrázku 3.2 (v popředí) představuje nejmenší možnou úroveň zoomu, celá oblast mapy je v takovém případě pokryta jednou dlaždicí. Maximální úroveň zoomu a nejdetailnější dlaždice jsou znázorněny v pozadí obrázku 3.2.

Ukázka URL pro GetTile:

```
<ServiceRoot>?SERVICE=WMTS &request=GetTile&version=1.0.0&
    Layer=Countries&style=default&format=image/png&
    TileMatrixSet=WholeWorld_CRS_84&TileMatrix=10m&TileRow=1&
    TileCol=3
```



Obrázek 3.2: WMTS TileMatrix (úrovně zoomu)[1]



Obrázek 3.3: Dlaždicový prostor WMTS TileMatrix [1]

3.2.2 GetCapabilities request

Formát dotazu je stejný jako u WMS. Vrátí XML dokument s informacemi o WMST serveru (uspořádání dlaždic, možné projekce, vrstvy atd.).

3.2.3 Rest request

Na základě metadat v ServiceMetadata.xml standardizuje HTTP GET požadavek pro snadné dotazování dlaždic. Dotaz nemá tvar KVP, ale je definován pomocí URI šablony. V mapových API je to nejpoužívanější metoda pro získávání obrázkových dlaždic.

Rest request tak může vypadat v závislosti na URI šabloně následovně:

```
<ServiceRoot>/{TileMatrix}/{TileRow}/{TileCol}[.Format]
```

Kapitola 4

Zdroje dat dostupné na webu

V úvodu této kapitoy je vysvětlení pojmu mashup aplikace. Následuje popis vybraných zdrojů dat dostupných na webu, které mohou být pro mashup aplikace použity. Kapitola je zaměřena především na zdroje geografcikých dat, které budou využity jako mapové podklady pro vytvořenou webovou aplikaci. Součástí je i popis volně přístupných dat z Českého statistického úřadu a portálu Europe's public data, které by mohli sloužit jako další zdroje dat pro mashup aplikace.

4.1 Mashup aplikace

Za mashup aplikaci (z anglického mashup = míchat) se označuje webová služba, nebo stránka, která kombinuje více datových online zdrojů zpravidla se svými vlastními daty. Získaná data následně vhodně reprezentuje, čímž vytváří na data nový pohled. V současné době mnoho webových služeb uvolňuje svá API, která vývoj mashup aplikací zjednodušují. Typickým příkladem jsou mapová API, která umožňují zobrazovat data na mapě na základě GPS souřadnic.

4.2 OpenStreetMap

OpenStreetMap [9] je open source projekt zabývající se tvorbou volně dostupných geografických dat a mapových podkladů. Veškerá data jsou poskytována pod licencí Open Database Licence (ODL) [10], která umožňuje využívat data zdarma i pro komerční účely. Jedinou nutnou podmínkou je zobrazovat v rohu mapy copyright OpenStreetMaps. Spojením geografických dat s mapovými podklady se dají vytvářet takzvané topografické mapy. Geografická data se použijí jako podvrstva zobrazované mapy a výsledkem může být např. silniční či železniční síť, turistické nebo cyklistické stezky, památky a jiné. Od roku 2008 je možné ukládat data do GPS zařízení a používat je pro navigaci.

4.2.1 Zdroje dat

Projekt OpenStreetMap funguje na obdobném principu jako Wikipedie. Daty z GPS přijímače, či jinými daty, která nejsou v rozboru s licencí OpenStreetMap, do něj může dobrovolně přispět kdokoli. Do projektu přispívají zejména:

- **Dobrovolníci** - Lidé mapující terén GPS přijímači. Svá data zpracují na počítači a nahrají do databáze OpenStreetMap.
- **Veřejné zdroje** - Vládní organizace, které svá data, či mapy uvolňují pod licencí kompatibilní licencí s OpenStreetMap. Příkladem poskytovaných dat jsou letecké či satelitní snímky, katastrální a silniční mapy apod.
- **Komerční zdroje** - firmy, které poskytují svá data (mapy, letecké snímky, silniční sítě, ...).

4.2.2 Pokrytí a formát dat

Geografická data OpenStreetMap pokrývají celý svět. Nejvíce je pokryta Evropa a USA.

Formát geografických dat OpenStreetMap je založen na XML. Důležitou vlastností je použitý souřadnicový systém - WGS 84[1] a Mercatorova projekce[2] použitá pro zobrazení dat. Souřadnicový systém dat musí být shodný se souřadnicovým systémem mapového serveru, případně se musí provádět přepočet. Jinak bude docházet k chybné reprezentaci dat. Všechny dlaždice jsou ve formátu PNG a mají rozměr 256 x 256 pixelů. Číslované jsou od 0, která je umístěna v levém horním rohu mapy. Pojmenování dlaždic je $\{z\}/\{x\}/\{y\}.png$, kde z je zoom, x číslo soupce a y číslo řádku v dlaždicovém prostoru.

4.3 Thunderforest

Thunderforest [13] v současné době nabízí zdarma tři různé mapové podklady (viz obrázek 4.1), distribuované pod licencí CC-BY-SA 2.0 (Creative Commons). Komerční použití je povoleno. Je však nutné brát v úvahu to, že tyto mapové podklady vznikly z geografických dat OpenStreetMap. Proto je nutné při jejich používání dbát i na licenci pod kterou distribuuje data OpenStreetMap. Při používání těchto map musíme zobrazoval copyright obou zdrojů (Thunderforest i OpenStreetMap).

Všechny dlaždice jsou ve formátu PNG a mají rozměr 256 x 256 pixelů. Číslované jsou od 0, která je umístěna v levém horním rohu mapy. Pojmenování dlaždic je $\{z\}/\{x\}/\{y\}.png$, kde z je zoom, x číslo soupce a y číslo řádku v dlaždicovém prostoru.

[1] World Geodetic System 1984 - geodetický standard vydaný ministerstvem obrany USA.
[2] Kartografické zobrazení plochy elipsoidu do roviny

4.3.1 OpenCycleMap

Mapa určená především pro cyklisty, obsahuje vrstevnice, detailní vyobrazení cyklostezek, lesních cest, silnic, zvýrazněné zalesněné plochy, atd.

URL šablona[3]:

```
http://[abc].tile.opencyclemap.org/cycle/${z}/${x}/${y}.png
```

4.3.2 Transport

Mapa zaměřená na dopravní informace jako jsou železniční a silniční sítě, MHD a metro. V Plzni jsou při dostatečném zvětšení i trasy a zastávky MHD, včetně čísel linek, které na nich jezdí.

URL šablona je následující:

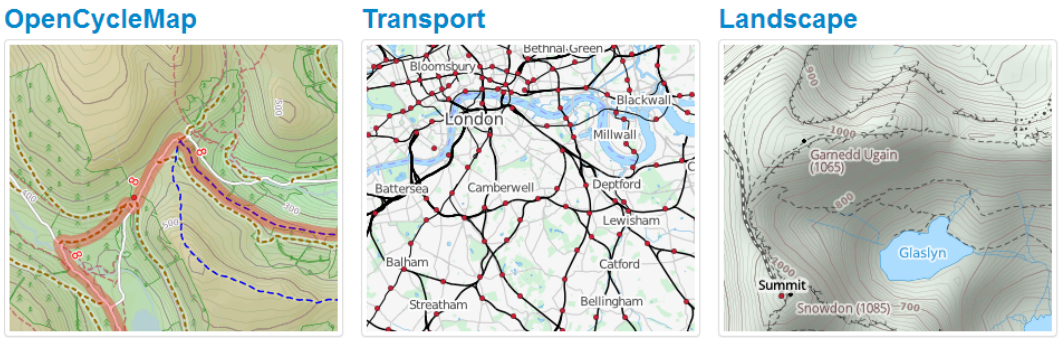
```
http://[abc].tile2.opencyclemap.org/transport
/${z}/${x}/${y}.png
```

4.3.3 Landscape

Mapa zaměřená hlavně pro outdoorové aktivity. Obsahuje údaje o terénu jako jsou vrstevnice včetně údajů o nadmořské výšce, lesní cesty, zvýrazněné lesy, popisy a názvy vodních toků apod.

URL šablona je následující:

```
http://[abc].tile3.opencyclemap.org/landscape
/${z}/${x}/${y}.png
```



Obrázek 4.1: Ukázky mapových podkladů z Thunderforest [13]

[3] [abc] označují dostupné subdomény (z URL je lze i vypustit) je následující

4.4 Český statistický úřad

Český statistický úřad [3] (ČSÚ) poskytuje statistické údaje České Republiky z různých odvětví a kategorií. Vize ČTK byla tato data zahrnout do mashup aplikace. Například propojit události zobrazené na mapě s údaji o velikosti sídla.

Forma, jakou data ČSÚ poskytuje, je vhodná spíše pro ruční procházení, než automatický sběr dat. Uživatel si pohodlně dokáže zobrazit co ho zajímá. Nicméně data poskytovaná na stránkách ČSÚ jsou směsí různých formátů (PDF dokumenty, dokumenty MS Excel, html tabulky či výpisy tabulek z *Veřejné databáze*), nevhodných pro mashup aplikace. Přehled těchto dat lze nalézt v záložce *Zdroje dat* na webu ČSÚ. U výpisů z tabulek Veřejné databáze je většinout k dispozici i vizualizace dat ve formě grafů.

Veřejná databáze obsahuje statistické údaje z databáze ČSÚ přístupné pomocí webového prohlížeče. V současné době umožňuje zobrazovat pouze celé tabulky a některé z těchto tabulek lze i graficky zobrazit ve formě grafu nebo geografického rozložení. U všech tabulek je odkaz na průvodce, který umožňuje export dat do formátu XSL nebo XML. Toto je podle mého názoru nejlepší způsob, jak by se aktuální data ČSÚ případně dala využít pro mashup aplikace. Bylo by ale nutné se podrobně seznámit s XML formátem, ve kterých jsou data exportována a strukturou Veřejné databáze. Metadata z exportu totiž obsahují i identifikátory na číselníky jiných tabulek.

4.5 Europe's Public Data

Europe's Public Data [4] (EPD) je datový portál, který poskytuje datové množiny v nejrůznějších formátech ze subjeků (lidí, firem, státních institucí) z celé Evropy. Datové množiny jsou poměrně přehledně rozčleněny do kategorií, které lze dále filtrovat podle dalších parametrů: skupina, formát dat a tag. Každá množina obsahuje množství metadat podle toho, jaké položky při jeho přidávání na portál uživatel vyplnil.

EPD se na první pohled jeví jako dobrý portál, kde jsou na jednom místě dostupné různé typy dat a to i pro Českou Republiku (např. statistické údaje z ČSÚ, Ministerstva financí a dalších ministerstvech). Při podrobnějším zkoumání bylo zjištěno, že většina souborů nejde otevřít a nejčastější příčinou byla chyba HTTP 404, (soubor nenalezen). Chyba vznikla tím, že daný datový zdroj (soubor) na který odkazuje záznam na stránkách EPD byl smazán, přejmenován, či přesunut a nebyl aktualizován jeho záznam v EPD portálu.

Kapitola 5

Nástroje pro vizualizaci dat

Tato kapitola obsahuje popis vlastností nalezených volně dostupných nástrojů pro vizualizaci dat a jejich srovnání. Zaměřena je na nástroje pro geografickou a časovou vizualizaci dat.

5.1 Google Maps

Google Maps API [16] je patrně nejznámější mapové API. Poskytuje pokročilé funkce pro plánování tras (pro pěší turistiku, cykloturistiku, dopravu, ...), pohledy z ulice, 3D budovy, počasí a další. Neustále se vyvíjí. Kromě základních funkcí potřebných pro vizualizaci GPS dat (značky, body, křivky, mnohoúhelníky), umožňuje definovat shlukování značek na mapě přímo pomocí API, podle jejich vzájemné vzdálenosti a úrovně zoomu. Nabízí také funkce umožňující geokódování (proces kdy se konkrétní adrese přiřadí GPS souřadnice) a reverzní geokódování (proces opačný).

V současné době je Google Maps API ve verzi 3, která není zpětně kompatibilní s předchozími verzemi. Do roku 2011 bylo Google Maps API poskytováno zcela zdarma. Nyní je zdarma pouze pro nekomerční použití, navíc s některými omezeními. Webové stránky a aplikace využívající Google Maps API zdarma, jsou omezeny počtem vygenerovaných událostí tzv. map loads [1]. Událost však není definována jen jako načtení mapy. Celkově lze za den vygenerovat 25 000 událostí. Překročení limitu nemá na funkčnost aplikace žádný vliv, pokud se tak neděje 90 po sobě následujících dní. Poté přijde od společnosti Google upozornění a bude třeba danou situaci řešit. Z licence plyne, že aplikace využívající Google Maps API nesmí být používány v rámci intranetu, za firewallem, nebo poskytovány uživatelům internetu, kteří se pro přístup musí registrovat za poplatek. Samotné omezení přístupu uživatelů prostřednictvím registrace a přihlašování uživatelů v rozporu s licenčním ujednání není. Pro komerční použití je možno zakoupit licenci Google Maps API for Bussines. Cena však začíná na 10 000 $ za rok, v závislosti na počtu zobrazení map.

Google Maps poskytuje tři základní mapové podklady: ulice, satelit, terén. API je možné použít i s jinými mapami. Poplatky a omezení však nesouvisí s použitím

[1] https://developers.google.com/maps/faq#usage_mapload

map od Google, ale na využívání API.

Následuje přehled hlavních vlastností pro Google Maps API:

- **Licence:** Google Maps[2]
- **Cena:** Zdarma pro nekomerční použití
- **Programovací jazyk:** JavaScript
- **Dokumentace:** Ano
- **Ukázky kódu, tutoriály:** Ano
- **Shlukování značek:** Ano
- **Geokódování:** Ano
- **Reverzní geokódování:** Ano
- **Funkční omezení:** Ano - počet zobrazení za den
- **Globální mapové podklady:** Ano

5.2 Mapy API

Mapy API [18] je velice přehledné API od české firmy Seznam.cz. Podporuje geokódování, reverzní geokódování i plánování tras. Dále zejména pro ČR navíc nabízí turistické mapy, cyklomapy, letecké snímky a další funkce. Vytváření vlastních aplikací je zcela intuitivní. Pro vizualizaci GPS dat (značky, body, křivky, mnohoúhelníky) obsahuje dostatek funkcí, ale narozdíl od Google Maps neumožňuje vytvářet shluky značek. Podobná funkce v API není záměrně, protože jeho vývojáři tvrdí, že shlukování se má provádět na straně serveru, nikoli na straně klienta. Je napsané v JavaScriptu a interně používá framework JAK [3].

Použití je zcela zdarma, uživatelé internetu však nesmí být nijak omezeni v přístupu na stránky, kde je API použito. Povoleno je pouze přihlašování uživatelů, avšak nikoliv například zpoplatněný přístup uživatelů, nebo provozování API v rámci intranetu či za firewallem.

V současné době se připravuje změna licenčního ujednání, kde by mělo být zrušeno zmíněné omezení a není ani vyloučena možnost domluvy přímo s firmou Mapy.cz. Výhodou je existence oficiálního fóra API v českém jazyce. Nevýhodou je, že mapové poklady nepokrývají celý svět, jsou zaměřeny především na ČR a Evropu.

Mapové podklady se mohou použít i s jiným API, musí být však dodrženy licenční podmínky Mapy API a na mapách zobrazeno nezměněné logo a copyright Mapy.cz.

[2] https://developers.google.com/maps/terms?hl=cs

[3] Javascriptová knihovna http://jak.seznam.cz/ - open source projekt, který velice zjednodušuje práci v prostředí jazyka JavaScript

Následuje přehled hlavních vlastností pro Mapy API:

- **Licence:** Mapy API[4]
- **Cena:** Zdarma
- **Programovací jazyk:** JavaScript
- **Dokumentace:** Ano
- **Ukázky kódu, tutoriály:** Ano
- **Shlukování značek:** Ne
- **Geokódování:** Ano
- **Reverzní geokódování:** Ano
- **Funkční omezení:** Ne
- **Globální mapové podklady:** Ne

5.3 OpenLayers

OpenLayers [8] je JavaScriptové API, které je poskytováno zcela zdarma jako OpenSource pod licenci FreeBSD[8]. Kromě OpenSourcové licence je jeho další velkou výhodou to, že je velmi flexibilní a není závislé na poskytovateli map nebo technologii. To je možné změnit bez nutnosti přepisovat celý kód. Podporuje např. Google, Yahoo, Microsoft, WMS, ArcGIS Server, MapServer atd. Obsahuje standardní funkce pro vizualizaci dat (značky, body, křivky, mnohoúhelníky). Oproti Google Maps je však pomalejší a komplikovanější a není určené pro mobilní aplikce. Díky OpenSource licenci je možné upravovat zdrojové kódy a přidávat nové funkce či vlastnosti [11].

Následuje hlavní přehled vlastností OpenLayers:

- **Licence:** FreeBSD
- **Cena:** Zdarma
- **Programovací jazyk:** JavaScript
- **Dokumentace:** Ano
- **Ukázky kódu, tutoriály:** Ano
- **Shlukování značek:** Ne
- **Geokódování:** Ne

[4] http://api4.mapy.cz/

- **Reverzní geokódování:** Ne
- **Funkční omezení:** Ne
- **Globální mapové podklady:** Dle vybraného poskytovatele

5.4 Microsoft Bing

Bing Maps [19] představil Microsoft jako alternativu k Google Maps. Rovněž se jedná o API založené na JavaScriptu. Podmínky používání Bing Maps se liší v závislosti na konkrétní licenci. Na výběr je 7 různých druhů, které se liší cenou a funkčností. Některé licence jsou zdarma a bez omezení, podmínkou je ovšem nekomerční použití a zároveň veřejný přístup. Cena u komerční licence se odvíjí od počtu relací nebo transakcí za rok, počtu uživatelů a dalších parametrů.

Následuje hlavní přehled vlastností Microsoft Bing:

- **Licence:** Bing Maps[5]
- **Cena:** Zdarma pro nekomerční použití, pro komerční použití dle konkrétní licence
- **Programovací jazyk:** JavaScript
- **Dokumentace:** Ano
- **Ukázky kódu, tutoriály:** Ano
- **Shlukování značek:** Ne
- **Geokódování:** Ano
- **Reverzní geokódování:** Ano
- **Funkční omezení:** dle zvolené licence
- **Globální mapové podklady:** Ano

5.5 Leaflet

Leaflet [15] je JavaScriptové OpenSource API navržené pro desktopové a mobilní aplikace. Díky dobře navrženému desingu se s ním lehce pracuje a také nabízí velikou flexibilitu v možnosti nastavení mapových podkladů. K mapovým serverům se přistupuje pomocí definovaných šablon. Pokud dokážeme sestavit správnou URL pro konkrétní mapový server (např. Google Maps a pod.), neměl by být technický problém použít jeho mapy s Leaflet API. V případě využití Google Maps však nic

[5] http://www.microsoft.com/maps/product/licensing.aspx

nebrání tomu, aby nás poskytovatel mapového serveru zablokoval [11]. Například pokud zjistí porušení licenčních podmínek Google Maps, nebo překračování povoleného počtu požadavků na zobrazení za den. Proto by bylo vhodné vybrat mapový server, u kterého neporušíme licenční ujednání, či copyright (např. OpenStreetMap).

Dobře navržený datový model umožňuje jednoduše přidávat nové funkce, nebo vytvářet nové pluginy. Vybrané pluginy je možné nalézt na stránkách projektu[6] a jejich využíváním lze programování v Leaflet API ještě více zpříjemnit. Jedním z nich je plugin *Leaflet.markercluster*[7], který umožňuje vytvářet shluky značek přímo pomocí API na klientské straně. Shlukování je velmi kvalitní a po přesunutí kurzoru myši na značku shluku se zobrazí okolí z kterého je shluk vytvořen (viz obrázek 5.1). Na stránce autora je uvedeno, že plugin umožňuje shlukovat i 50000 značek v prohlížeči Google Chrome, v IE9 jsou při takovém počtu značek se shlukování problémy. Shlukování 50000 značek bylo prakticky vyzkoušeno v prohlížečích Mozilla Firefox a Google Chrome. Zobrazení bylo v pořádku, počáteční vytvoření shluků trvalo několik sekund.

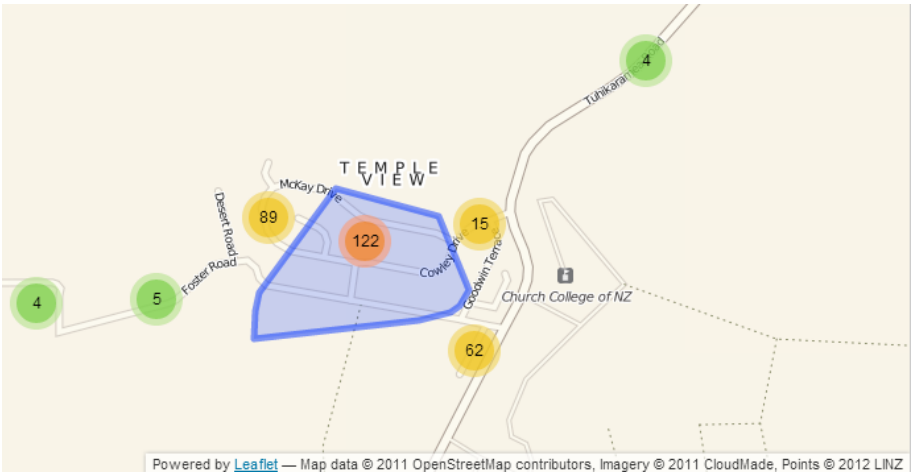
Následuje hlavní přehled vlastností API Leaflet:

- **Licence:** OpenSource
- **Cena:** Zdarma
- **Programovací jazyk:** JavaScript
- **Dokumentace:** Ano
- **Ukázky kódu, tutoriály:** Ano
- **Shlukování značek:** Ano - speciální plugin
- **Geokódování:** Ne
- **Reverzní geokódování:** Ne
- **Funkční omezení:** Ne
- **Globální mapové podklady:** Dle vybraného poskytovatele

5.6 SIMILE Widgets Timeline

SMILE Widgets je OpenSource projekt, který zdarma nabízí nástroje pro vizualizaci dat. Jedním z nich je Timeline [21] (časová řada) určená pro vizualizaci časových údajů, napsaná v JavaScriptu. Je tvořena horizontálním pásem (časovou osou), na které lze zobrazovat data jako události svázané s konkrétním časem, nebo s časovým intervalem. Časovou osu lze posunovat myší nebo kurzorovými klávesami. Při definici

[6] http://leafletjs.com/plugins.html
[7] https://github.com/danzel/Leaflet.markercluster



Obrázek 5.1: Shluky vytvořené pluginem Leaflet.markercluster

časové osy je možno specifikovat časové jednotky (*rok, měsíc, den, ...*), které slouží jako její měřítko. Zároveň lze vytvořit více časových řad s různým měřítkem, které lze mezi sebou synchronizovat. Aplikacei se poté zpřehlední a je možno se v ní snadněji navigovat. Použije se např. jedna časová osa pro roky a druhá jemnější pro dny. Timeline obsahuje také různé funkce pro vyhledávání událostí na časové ose, či posouvání osy na konkrétní datum a čas.

Hlavní nevýhodou Timeline je fakt, že v projektu již nikdo nepokračuje. Poslední uložená verze projektu je z roku 2009. Od té doby bylo objeveno mnoho chyb, které nikdo neopravuje. Na internetových diskusních fórech je sice možné nalézt řešení na konkrétní problémy, ale vytvářet aplikaci v nástroji s chybami a bez oficiální podpory, není vhodné řešení.

- **Licence:** OpenSource
- **Cena:** Zdarma
- **Programovací jazyk:** JavaScript
- **Výhody:** Možnost více synchornizovaných časových os
- **Nevýhody:** Čtyři roky stagnující projekt, obsahuje chyby, špatná dokumentace, většina funkcí bez popisu

5.7 Almende Timeline

Almende Timeline [14] je projekt distribuovaný pod licencí Apache Licence 2.0[8]. Je tedy možné využití zdarma i pro komerční účely. Časová osa Almende Timeline je napsaná v JavaScritpu, je horizontálně orientovaná a pohyb po ní se provádí tahem myší. Obsahuje i zoom, a tak uživateli umožňuje přibližovat či oddalovat události na časové ose. Při změně zoomu se plynule přepočítává a mění obsah časové osy i měřítka. Měřítko časové osy má rozsah od milisekund po roky. Zobrazovat lze události v konkrétním čase i v časovém intervalu. Timeline může být definována i jako editovatelná, čímž lze událostem měnit čas posunutím na časové ose. Je také možné editovat obsah událostí. Vzhled událostí lze jednoduše přizpůsobovat pomocí CSS stylů a k nim přidávat i obrázky. Mezi další užitečné věci patří seskupování událostí podle daného klíče, který je poté zobrazen na ose y. API také umožňuje shlukování velkého počtu událostí. Události se shlukují podle časové vzdálenosti a úrovně zoomu. Shlukování událostí je prozatím označené jako experimentální funkce.

Na stránkách projektu se uvádí, že Almende Timeline umožňuje bezchybně zobrazit 10 000 událostí.

- **Licence:** Apache Licence 2.0
- **Cena:** Zdarma
- **Programovací jazyk:** JavaScript
- **Výhody:** Projekt se neustále vyvíjí, zoom, snadné přizpůsobení událostí pomocí CSS, seskupování podle klíče, shlukování událostí na ose, přizpůsobování velikosti časové osy, editovatelný mód, velké množství ukázek, kvalitní dokumentace
- **Nevýhody:** Navigace - pouze jedna časová osa (měřítko se mění podle úrovně zoomu, proto se nelze vidět detaily událostí např. na úrovní dní a zároveň se rychle pohybovat o větší časové úseky např. roky)

5.8 Shrnutí vlastností

Analyzované nástroje pro geografickou vizualizaci jsou srovnány v tabulce 5.1. Tabulka 5.2 srovnává nástroje pro časovou vizualizaci.

Všechny analyzované nástroje měli jeden společný znak, byly naprogramovány v JavaScriptu.

[8] http://www.apache.org/licenses/LICENSE-2.0

API	Licence	Cena	Komerční využití zdarma a bez omezení	Shlukování značek	Funkční omezení	Globální mapové podklady
Google Maps	Google Maps	zdarma [9]	ne	ano	ano [9]	ano
Mapy API	Mapy API	zdarma	ne [10]	ne	ne	ne
OpenLayers	OpenSource	zdarma	ano	ne	ne	*[11]
Microsoft Bing	Bing Maps	zdarma [9]	ne	ne	*[12]	ano
Leaflet	OpenSource	zdarma	ano	ano (plugin)	ne	*[11]

Tabulka 5.1: Tabulka srovnání vlastností geografických API

API	Licence	Cena	Komerční využití zdarma a bez omezení	Shlukování událostí	Funkční omezení
SIMILE Widgets Timeline	OpenSource	zdarma	ano	ne	ano[13]
Almende Timeline	Apache Licence 2.0	zdarma	ano	ano	ne

Tabulka 5.2: Tabulka srovnání vlastností časových API

5.9 Výběr použitých prostředků

Na základě požadavků zadavatele měly vybrané nástroje splnit následující podmínky:

- nízká cena licence (nejlépe zdarma) pro komerční účely
- pokrytí celého světa (pro mapové nástroje)
- webová technologie

5.9.1 Geografické API - Leaflet

Z nalezených georafických nástrojů bylo možné pro komerční použití zdarma použít tři nástroje: Mapy API, OpenLayers a Leaflet.

[9] Pro nekomerční použití
[10] V současné době ne, ale pro rok 2013 se připravuje změna licenčního ujednání
[11] Nastavitelné mapové zdroje (takže ano např. při použití OpenStreetMap)
[12] Záleží na typu zvolené licence
[13] Ukončený vývoj, obsahuje množství neopravených chyb

Mapy API (viz kap. 5.2) nevyhovovalo ze dvou důvodů. Podle licenčního ujednání totiž aplikace s Mapy API musí být veřejně přístupná, což částečně znemožňuje komerční využití (např. přístup pouze registrovaných uživatelů, kteří si službu zakoupí). Druhým důvodem je geografické pokrytí, které je zaměřené pouze na Evropu a blízké okolí.

Co se týče požadavků, je **OpenLayers** (viz kap. 5.3) vhodný kandidát. Je zdarma, bez jakýchkoliv omezení a záleží jen na uživateli, jaké mapové poskytovatele si zvolí. Zajistit tak globální pokrytí také není žádný problém. Pro OpenLayers existuje i velké množství tutoriálů, přesto jsem však nalezl vhodnější nástroj, který OpenLayers svými vlastnostmi předčil.

Leaflet (viz kap. 5.5) je moderní API, které je narozdíl od OpenLayers navržené i pro mobilní aplikace. V porovnání s OpenLayers je celkově rychlejší, obsahuje řadu užitečných pluginů, včetně pluginu pro shlukování značek. Má jednoznačně nejlepší design a práce s ním je intuitivní, není třeba zdlouhavě studovat žádné tutoriály. K programování stačí dokumentace, která je přehledně zpracovaná.

5.9.2 Časová řada - Almende Timeline

Nástroje pro vizualizaci časových řad, které by bylo možné použít zdarma pro komerční účely, jsem nalezl pouze dva: Simile Widget Timeline a Almende Timeline. Oba tyto nástroje jsou velmi podoblé.

Simile Widget Timeline se již několik let nevyvíjí a obsahuje stále množství chyb, proto nebyl vybrán.

Pro realizaci aplikace zobrazující časová data jsem tedy vybral časovou řadu Almende Timeline, která umožňuje na časové ose scrollování, při kterém se měřítko osy přepočítává a je bez chyb. Dalším přínosem je podpora shlukování a seskupování událostí i kvalitně zpracovaná dokumentace.

Kapitola 6

Realizační část

Tato kapitola obsahuje návrh, popis použitých programovacích prostředků a vytvořených webových aplikací. Dále popis a vývojový diagram implementovaného shlukovacího algoritmu.

6.1 Návrh aplikace

Vstupem aplikace jsou XML soubory obsahující dokumenty a události ČTK ve formátu *NewsML* (viz kap. 2). Data budou zobrazována podle časových a geografikých (GPS souřadnice) metadat. Návrh aplikace je znázorněn na obrázku 6.1.

Dle požadavků zadavatele bude mítnavrhovaná aplikace strukturu klient-server.

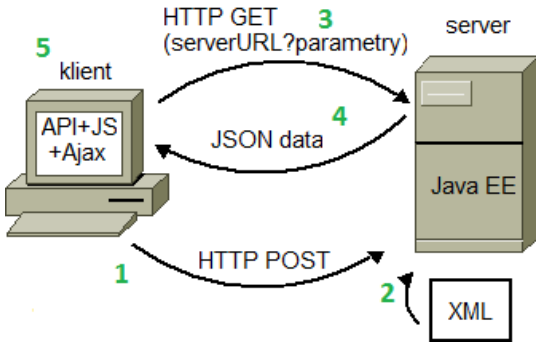
Serverová část se skládá z XML parserů pro načítání vstupních dat a řídícího servletu. XML parsery uchovávají rozparsované XML soubory v datové struktuře složené ze tříd, které blízce kopírují formát *NewsML*. Řídící servlet reaguje na HTTP požadavky typu POST parsováním vybraného XML souboru a na požadavky typu GET odesláním požadovaných dat podle parametru URL ve formátu JSON (JavaScript Object Notation) 6.2.2 klientovi. Na serverové straně bude také implementován algoritmus pro shlukování dat, podle jejich vzájemné vzdálenosti dané GPS souřadnicemi.

Klientské části jsou ve všech případech tvořené HTML stránkou, knihovnou použitého API, Ajaxovou knihovnou a javascriptovými soubory s definicemi funkcí, které společně s API umožňují vizualizovat data získávaná asynchronně ze serveru.

6.1.1 Popis procesů

Zde je chronologicky uveden popis jednotlivých procesů navržené aplikace, která je znázorněna na obrázku 6.1. Zelená čísla na tomto obrázku odpovídají číslům procesů v následujícím výčtu. Symboly K a S v hranatých závorkách za číslem procesu udávají místo, kde daný proces probíhá (K = klient, S = server).

Veškerá data, která se odesílají ze serveru do klientské aplikace jsou ve formátu JSON.



Obrázek 6.1: Návrh aplikace

1. [**K**] Výběr API a zdrojových dat na úvodní stránce - požadavek na server typu POST
2. [**S**] Zpracování požadavku typu POST - vytvoření *session*, parsování XML dat a přesměrování na stránku s konkrétním API vybraném v procesu (1)
3. [**K**] Zobrazení stránky s vybraným API, odeslání požadavku typu GET pro získání dat na server
4. [**S**] Zpracování požadavku typu GET, odeslání požadovaných dat klientovi
5. [**K**] Přijem a zobrazení dat

Za procesem (5), mohou od klienta následovat další požadavky typu GET. Pokud nevypšel timeout *session*, provedou se po odeslání takového požadavku opět procesy 4) a 5). V opačném případě server přesměruje klienta zpět na proces 1).

6.2 Použité programovací prostředky

Aplikace bude vytvořena v programovacím jazyce Java JDK 1.6, JavaScriptu, a javascriptových API Leaflet (viz 5.5) a Almende Timeline (viz 5.7). Jako webový server bude použit Apache Tomcat 6.0.18. Správnost zobrazení výsledků bude otestována v prohlížečích Internet Explorer 8, Google Chrome verze 26.0.1410.64 m, Opera 10.01 a Mozilla Firefox 3.6.11.

Následující podkapitoly obsahují krátký popis knihoven, které jsem v naprogramovaných aplikacích využil.

6.2.1 JQuery

JQuery [5] je OpenSource JavaScriptová knihovna zapouzdřující mnoho funkcí jako např. manipulování s HTML dokumentem, procházení DOM (Document Object

Model), zpracování událostí a hlavně Ajax (Asynchronous JavaScript and XML). Ve vytvořených aplikacích bude knihovna JQuery použita právě kvůli Ajaxu.

Ajax umožuje interakci webových aplikací bez nutnosti znovunačtení stránky.

6.2.2 JSON

JSON (JavaScript Object Notation) [6] je nezávislý textový formát pro výměnu dat vycházející z konvecní JavaScriptu. Je lehce čitelný. Je založen na dvou strukturách: 1) kolekci párů *název : hodnota* (např. objekty v jazyce Java); 2) tříděném seznamu *hodnot* (ve většině jazyků realizován jako pole).

Jedná se o univerzální datové struktury a v podstatě všechny moderní programovací jazyky je v nějaké formě podporují. Je tedy logické, aby na nich byl založen i na jazyce nezávislý výměnný formát [6].

Objekt je v JSON uvozen znakem **{** a končí znakem **}**. Uvnitř objektu je množina párů název**:***hodnota* a jednotlivé páry jsou od sebe oděleny čárkou.

Pole v JSON začíná znakem [a končí znakem], *hodnoty* jsou od sebe odděleny čárkou.

Hodnota ve strukturách JSON může být tvořena následujícími typy: *řetězec* (uzavřený uvozovkami), *číslo*, *objekt*, *pole*, *true*, *false* a *null*. Hodnoty do sebe mohou být vnořovány.

Na následujícím příkladu je struktura objektu v Javě a dále jeho reprezentace v JSON formátu:

```
public class MarkBean implements Serializable {

  private double latitude;
  private double longitude;
  private String city;
  private int itemCounter;
  private boolean cluster;
  private Map<String, Integer> category;
.
.
.
}
```

Ekvivalentní zápis třídy MarkBean naplněné daty v JSON:

```
{"latitude":50.03915,"longitude":16.039801,"city":"Ostřetín
  ","itemCounter":1,"cluster":false,"category":{"zak":1,"
  dpr":1}}
```

JSON je použit v aplikacích pro přenos dat ze serveru ke klientovi. V JavaScriptu není třeba JSON nikterak parsovat a je možné přistupovat k jeho hodnotám přímo

pomocí tečkové notace. Oproti XML je také méně paměťově náročný. Z těchto důvodu byl upřednostněn před XML formátem.

6.2.3 Gson

Gson [17] je OpenSource Java knihovna ke konverzi Java objektů do JSON formátu a naopak. Gson nepotřebuje v java třídách žádné speciální anotace a dokáže konvertovat základní datové typy i java kolekce a generické typy.

Na serverové straně je použit Gson pro konverzi všech odpovědí těsně před jejich odesláním. Veškeré odpovědi ze serveru jsou tak ve formátu JSON.

Ukázka použití:

```
int[] array = {1, 2, 3};
// hotota řetězce jsonArray bude: [1, 2, 3]
String jsonArray = new Gson().toJson(array);
```

6.2.4 JAK

JAK (JAvaScriptová Knihovna) [20] je OpenSource knihovna od firmy Seznam.cz, která se používá ve většině jejich aplikací. Svými funkcemi, které zapouzdřují složitější konstrukce v JavaScriptu usnadňuje psaní skriptů a činí kód přehlednějším.

JAK řeší práci s DOM a HTML událostmi, Ajaxem a rozšiřuje možnosti JavaScriptu o pokročilé možnosti OOP. Na webových stránkách projektu je k dispozici množství užitečných widgetů (miniaplikací) a utilit vytvořených s použitím knihovny JAK. Jedná senapříklad o utility pro práci s grafikou, XML nebo widgety jako kalendář, Drag & Drop (táhni a pusť), různé ovládací prvky a další.

JAK je použit v aplikacích pro geografickou vizualizaci na procházení DOM (Document Object Model).

6.3 Serverová část aplikace

V této kapitole je popsána struktura serverové části aplikace (jednotlivé package a java třídy s popisem), typy požadavků na které řídící servlet reaguje a typy odpovědí. Dále je v této části uveden popis algoritmu použitého pro shlukování značek spolu s použitou metrikou pro výpočet vzdáleností značek.

Značkou jsou dále v tomto textu myšlena data se strukturou odpovídající třídě *MarkBean* (viz kap. 6.3.3). A to na serverové straně v podobě java třídy i na straně klienta, který používá formát JSON.

6.3.1 UML dialgram

UML diagram serverové části je obsažen v příloze C na obrázku C.1. Originál obrázku ve větším rozlišení je na přiloženém DVD.

Zelený obdelník v pravém horním rohu diagramu reprezentuje třídy z balíku *zcu.xml.objects* (viz kap. 6.3.5), které jsem kvůli přehlednosti do diagramu nezakreslil. Popis jednotlivých tříd zobrazených na diagramu je rozepsán v následujících podkapitolách.

6.3.2 Package zcu.map

GetData.java

Řídící J2EE servlet oddědený od třídy *HttpServlet*. Servlet je v aplikaci namapovaný na */GetData*. Celá URL servletu má tedy tvar:

```
server_url:port/WebApplicationCTK/GetData
```

V metodě ***doPost(...)*** reaguje na *HTTP* požadavky typu *POST* s parametry *api* a *data*, které určují jaké API a která data se mají použít pro zobrazení. Po přijmutí požadavku se vytvoří nová session pokud neexistuje a rozparsuje se XML soubor se vstupními daty. Název adresáře kde jsou uloženy vstupní soubory je definován v souboru *WEB-INF/web.xml* vlastností *document_root*. Po načtení vstupního souboru dojde k přesměrování na dané API, např. Leaflet.

Metoda ***doGet(...)*** reaguje na *HTTP* požadavky typu *GET*, které přichází z jednotlivých API a vrací data ve formátu *JSON* viz. 6.2.2. K tomu je nutné nastavit typ odpovědi následovně:

```
response.setContentType("text/html;charset=UTF-8");
response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");
```

Možné parametry pro požadavky typu *GET* a odpovědi[1] na ně:

- *markers=list* - vrátí seznam všech značek *java.util.List<MarkBean>*.
- *markers=map* - vrátí všechny značky jako mapu (= redukce značek se stejnými GPS souřadnicemi), kde klíč tvoří GPS souřadnice[3].
- *markers=map&year=rok&month=mesic* - stejně jako v *markers=map* vrátí značky jako mapu, pouze s tím rozdílem, že nevrací všechny značky, ale značky které mají v datumu obsažen stejný rok a stejný nebo dřívější měsíc.
- *markers=clusters&zoom=cele_cislo*[2] - aplikuje na značky shlukovací algoritmus a vrátí seznam značek *java.util.List<MarkBean>*, ve kterém jsou nashlukované značky nahrazeny značkami s GPS souřadnicemi vypočtených centroidů.

[1] Všechny odpovědi jsou před odesláním konvertovány do formátu *JSON*. *MarkBean, MarkCardBean* a *ObjectBean* jsou obalovací třídy popsané v 6.3.3

[2] Úroveň zoomu

- *markers=markInfo&coords=souradnice*[3] - vrátí seznam s informacemi o značkách na daných GPS souřadnicích.
- *show=timeline&coords=souradnice*[3] - vrátí seznam všech položek ze vstupního XML na daných souřadnicích obalený do *ObjectBean*.
- *show=timeline* - vrátí seznam všech položek ze vstupního XML obalený do *ObjectBean*.
- *show=id* - vrátí informace o položce ze vstupního XML s konkrétním *id* obalený do *ObjectBean*.

6.3.3 Package zcu.map.beans

Package *zcu.map.beans* obsahuje obalovací třídy pro data odesílané v *HTTP* odpovědích klientovi. Jejich použitím se snižuje celkový objem přenášených dat, protože obsahují pouze nejnutnější informace o daných položkách a zároveň se zpřehledňuje implementace na klientské straně.

MarkBean.java

Třída zapouzdřující nejnutnější metadata o značkách pro jejich zobrazování na mapě.

Atributy této třídy jsou:

- *latitude* - zeměpisná šířka
- *longitude* - zeměpisná délka
- *city* - název města
- *itemCounter* - počet položek, které značka reprezentuje
- *cluster* - příznak, zda se jedná o shluk více značek vytvořený shlukovacím algoritmem 6.3.7 či nikoli
- *category* - kategorie, které značka zastupuje včetně jejich četnosti

MarkCardBean.java

Třída zapouzdřující metadata pro vyskakovací okna, které se zobrazují po kliknutí na značku na mapě.

Atributy této třídy jsou:

- *documentType* - typ dokumentu
- *dateId* - datum, odpovídá hodnotě atributu *dateId* v případě že se jedná o dokumety, nebo *dateStart* v případě událostí

[3] GPS souřadnice ve tvaru: *latitude;longitude*

- *publicIdentifier* - unikátní identifikátor
- *title* - titulek, rovněž závisí na tom, zda se jedná o incidenty nebo dokumenty, v případě incidentů odpovídá atributu *comment*, u dokumentů odpovídá atributu *headline* pokud je definován, či atributu *caption* pokud *headline* definován není

6.3.4 Package zcu.map.util

Cluster.java

Třída *Cluster.java* slouží ke shlukování značek podle jejich vzájemné vzdálenosti a úrovně zoomu. Detailní popis použitého algoritmu pro shlukování je v kapitole 6.3.7.

Třída obsahuje tyto veřejné metody:

- *Cluster(Map<String, MarkBean>marks, int zoom)* - konstruktor třídy, který podle parametrů inizializuje proměnné pro výpočet shluků.
- *createClusters()* - metoda, která provede nashlukování značek podle algoritmu 6.3.7 a následně zavolá privátní metodu pro výpočet cetroidů viz 6.3.8. Výsledek obou dvou kroků algoritmu (nashlukování značek a následné vytvoření značek ze shluků) je vrácen jako *java.util.List<MarkBean>*, kde každý shluk je tvořen jednou značkou s příznakem *cluster* a čítačem značek které reprezentuje.

DocsParser.java

XML parser implementující rozhraní *IParser.java*. Slouží k parsování *dokumentů* se strukturou viz 2.1.

IParser.java

Rozhraní pro XML parsery. Všechny návratové hodnoty metod jsou z package *zcu.map.beans* (viz 6.3.3).

- *getItemByID(String id)* - vrací kompletní metadata záznamu s konkrétním *id* z parsovaného XML souboru.

  Metadata vrací jako objekt *ObjectBean*.
- *getItems()* - vrací všechna metadata z parsovaného XML souboru. Metadata vrací jako objekt *ObjectBean*.
- *getItems(Double lat, Double lng)* - vrací metadata všech záznamů z parsovaného XML souboru na konkrétních GPS souřadnicích určených parametry *lat* (latitude) a *lng* (longitude).

  Metadata vrací jako objekt *ObjectBean*.

- *getMarkInfo(Double lat, Double lng)* - vrací základní informace pro vizualizaci záznamů na konkrétních GPS souřadnicích z parsovaného XML souboru. GPS souřadnice jsou určeny parametry *lat* (latitude) a *lng* (longitude).

  Vrací *java.util.List<MarkCardBean>*.

- *getMarksAsList()* - vrací základní informace pro vizualizaci záznamů z parsovaného XML souboru jako seznam.

  Vrací *java.util.List<MarkBean>*.

- *getMarksAsMap()* - vrací základní informace pro vizualizaci záznamů z parsovaného XML souboru ve formě *hašovací tabulky* [4], jejíž klíčem jsou GPS souřadnice. Tím se redukují záznamy se stejnými GPS souřadnicemi.

  Vrací *java.util.Map<MarkBean>*.

- *getMarksAsMap(String year, String month)* - obdoba předchozí metody s tím rozdílem, že nevrací informace o všech záznamech. Záznamy musí v datu obsahovat stejný rok (parametr *year*) a stejný či libovolný předchozí měsíc (parametr *month*).

  Vrací *java.util.Map<MarkBean>*.

- *parse()* - metoda která rozparsuje vstupní XML soubor.

IncsParser.java

XML parser implementující rozhraní *IParser.java*. Slouží k parsování *událostí* se strukturou viz 2.2.

ValueComparator.java

Komparátor určený pro řazení kategorií podle četnosti. Kategorie jsou u značek či shluků ukládány jako *java.util.Map<String, Integer>*, kde klíč tvoří zkratka kategorie a hodnu její četnost. Standardně se totiž kolekce *java.util.Map<key, value>* řadí podle klíče, nikoli hodnoty. S třídou *ValueComparator.java* se kolekce dá seřadit podle hodnot.

Ukázka použití *ValueComparator* pro seřazení kategorií uložených v *java.util.Map<String, Integer>* (proměnná *category*):

```
ValueComparator vc = new ValueComparator(category);
Map<String, Integer> sorted = new TreeMap<String,
    Integer>(vc);
sorted.putAll(category);
category = sorted;
sorted = null;
```

[4] datová struktura která asociuje hašovací klíče s odpovídajícími hodnotami

6.3.5 Package zcu.map.xml.objects

Package *zcu.map.xml.objects* obsahuje třídy, které blízce kopírují formát *NewsML* a uchovávají tak důležité metadata. Během parsování dokumentů nebo událostí jsou rozparsované atributy a elementy ukládány do struktur definovaných v package *zcu.map.xml.object*.

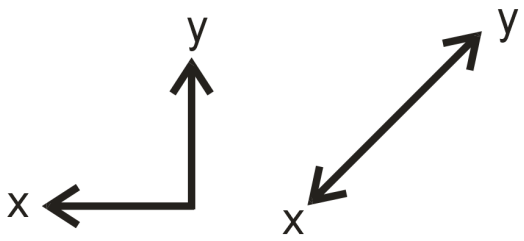
6.3.6 Manhattan vzdálenost

Manhattan vzdálenost [7] je metrika pro určení vzdálenost mezi dvěmi body. Vzdálenost je měřena podél os v pravém úhlu. Rozdíl mezi Manhattan a Euklidovskou vzdáleností znázorňuje obrázek 6.2.

Manhattan vzdálenost d mezi body X a Y ($X = [x_1, x_2]$, $Y = [y_1, y_2]$) se vypočte podle vzorce 6.1:

$$d = |x_1 - y_1| + |x_2 - y_2| \tag{6.1}$$

Ve shlukovacím algoritmu je použita tato metrika pro menší výpočetní náročnost. Použití jiné metriky např. Euklidovské vzdálenosti by velmi nepříznivě ovlivnilo čas výpočtu.



Obrázek 6.2: Rozdíl mezi Manhattan (vpravo) a Euklidovskou (vlevo) vzdáleností

6.3.7 Shlukovací algoritmus

Umístění velkého počtu značek na mapu má dva zásadní problémy: 1) zobrazení by bylo nepřehledné; 2) práce s API by byla pomalá, nebo by se značky nedokázaly zobrazit vůbec. Z těchto důvodů je vhodné nevytvářet pro každý datový záznam značku, ale vytvořit na základě konkrétních parametrů shluky dat a data zobrazovat jako jednu značku.

Pro tento účel jsem modifikoval existující algoritmus [2] a využil ho pro shlukování dat. Data shlukuje podle jejich vzájemné vzdálenosti, která se dá určit z GPS souřadnic obsažených v metadatech.

Na obrázku 6.3 je princip shlukovacího algoritmu, který jsem použil pro shlukování geografických značek na serverové straně aplikace. Popis shlukovacího algoritmu je uveden v následujících dvou odstavcích.

Vstupem algoritmu je seznam značek s GPS souřadnicemi (zeměpisná šířka, zeměpisná délka) a úroveň zoomu. Výstupem algoritmu jsou dva seznamy. První seznam tvoří samostatné značky (*single_markers*), tj. značky, které nemohly být seskupeny, protože byly od ostatních značek příliš vzdáleny. Druhým seznamem je seznam shluků (*cluster_markers*). Shluk je zde tvořen seznamem značek, které mají být seskupeny, protože jejich vzájemná vzdálenost je menší než definovaná hranice. Pro výpočet vzájemné vzdálenosti mezi dvěmi značkami používám Manhattan vzdálenost.

Ze vstupního seznamu značek jsou postupně vyjímány značky dokud není prázdný. Po každém vyjmutí značky se porovnává její poloha s polohou všech zbývajích značkek ve vstupním seznamu a zároveň je vytvořen prázdný seznam shluků. Pokud je vzdálenost těchto značek menší než definovaná minimální vzdálenost (funkce závislá na úrovni zoomu), je právě porovnávaná značka ze vstupního seznamu značek odstraněna a je uložena do seznamu shluků. Následuje zpracování další značky ve vstupním seznamu. Když je původně vyjmutá značka porovnána se všemi zbývající značkami ve vstupním seznamu, zjistí se velikost vytvořeného seznamu shluků, a pokud je větší než nula, přidá se do něho tato značka a následně se celý tento seznam přidá do seznamu *cluster_markers*. V opačném případě se přidá do seznamu *single_markers*, kam se umísťují značky které nejsou součástí žádného shluku.

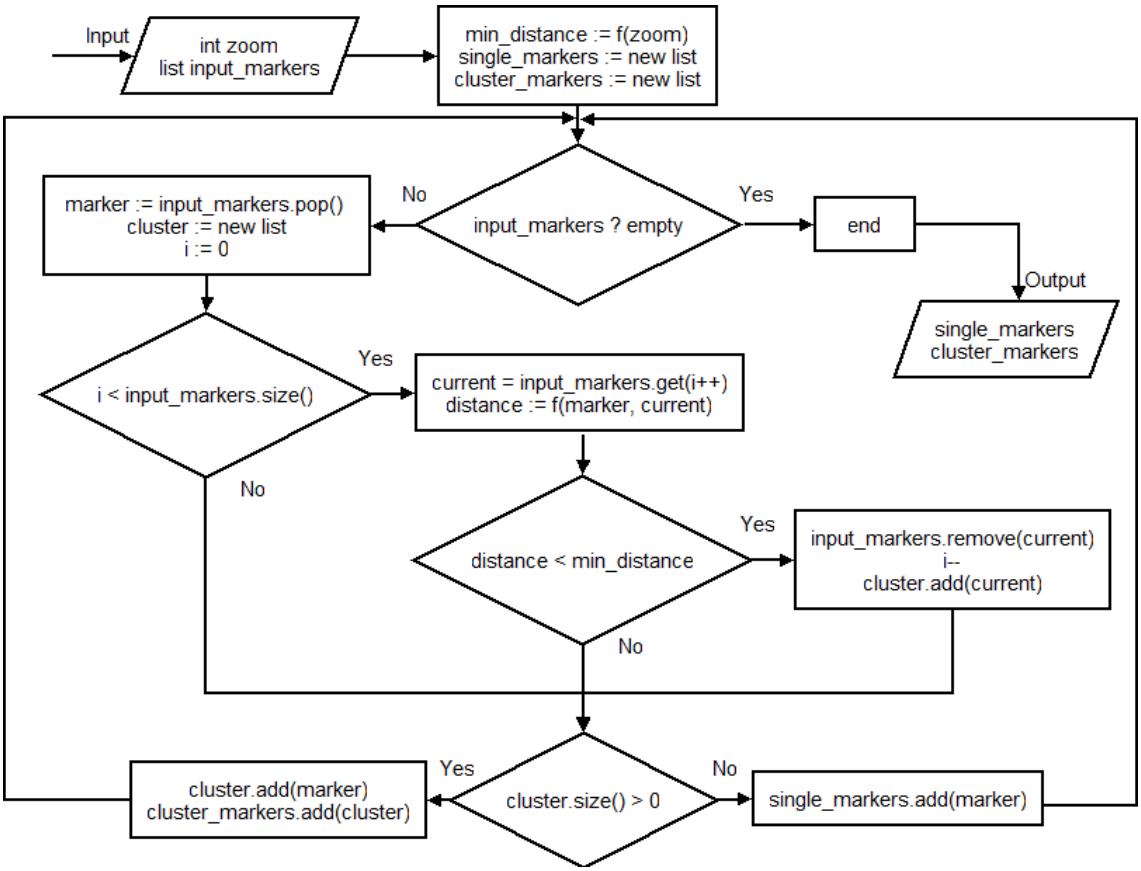
Po provedení algoritmu jsou původní data rozdělena ve dvou seznamech. Seznam samostatných značek (*single_markers*) je ihned možné zobrazit na mapě, protože každá položka seznamu v sobě obsahuje GPS souřadnice. Seznam shluků (*cluster_markers*) je však nutné ještě dále zpracovat, protože jeho položky neobsahují přímo GPS souřadnice shluku. Aby bylo možné shluky z tohoto seznamu správně zobrazit na mapě, musí se pro každý shluk vypočítat takzvaný centroid, více viz následující podkapitola.

6.3.8 Výpočet centroidu

Výpočet centroidu se provádí pro seznam značek, které reprezentují shluk. Každá značka v takovém seznamu obsahuje GPS souřadnice. Cílem algoritmu je z tohoto seznamu získat centroid neboli pozici (GPS souřadnice), pod kterými bude shluk zobrazen na mapě.

Centroid je v podstatě aritmetickým průměrem GPS souřadnic ve shluku. Na obrázku 6.4 je uveden vývojový diagram výpočtu centroidů.

Na konci zpracování každého shluku je vytvořena nová značka reprezentující shluk s vypočtenými GPS souřadnicemi, kterou lze již zobrazovat na mapě.



Obrázek 6.3: Vývojový diagram shlukovacího algoritmu

6.4 Klientská část - API Leaflet

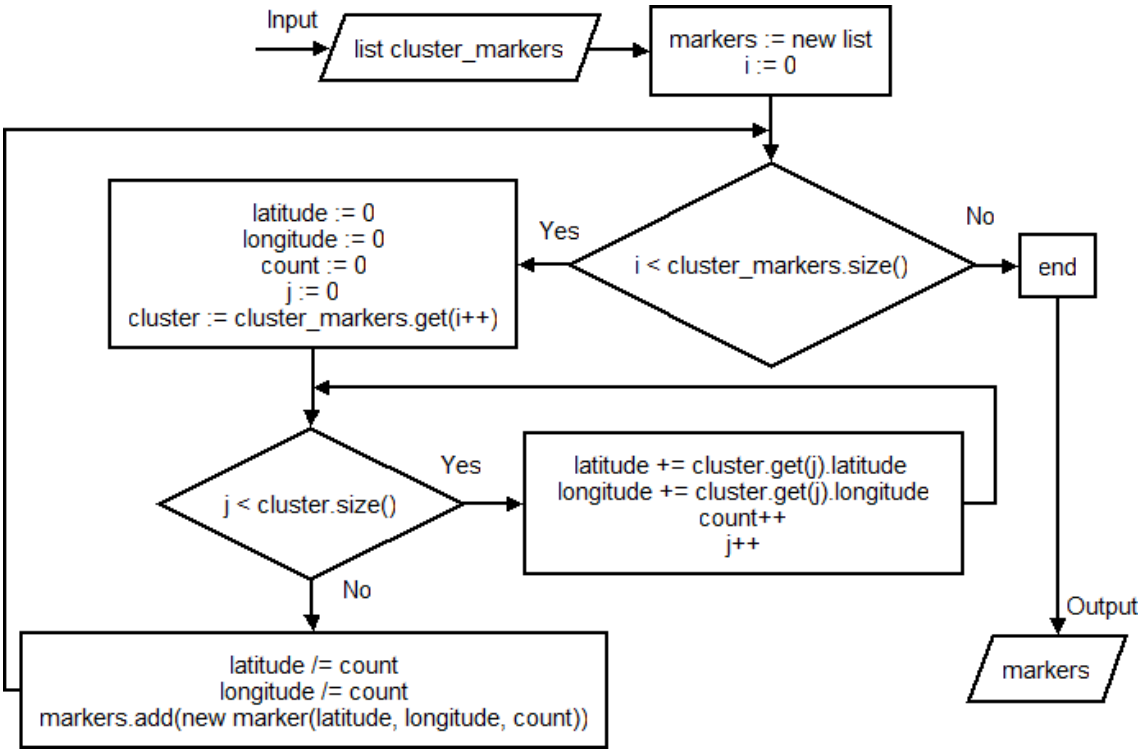
Tato kapitola popisuje první část klientské aplikace vytvořené pomocí *API Leaflet* (viz 5.5). Aplikace demostruje několik možností, jak zobrazovat data ČTK na základě GPS souřadnic.

Kvůli mapovým podkladům, které se získávají ze speciálních serverů (viz 3) je pro spuštění této aplikace aplikace nutné připojení k internetu.

6.4.1 Popis funkčnosti

Geografická vizualizace dat je přístupná pomocí stránky *WebApplicationCTK/leaflet/LEAFLET.html*. Její struktura je vidět na obrázku 6.5. Bližší popis jednotlivých prvků této stránky je v níže uvedeném výčtu, jehož body odpovídají očíslovaným prvkům z obrázku 6.5.

- **1** - Úroveň aktuálního zoomu (*DIV* element s *id="zoom"*).
- **2** - Formulář s *id="markers"* pro změnu zobrazení dat na mapě (viz následující podkapitola).

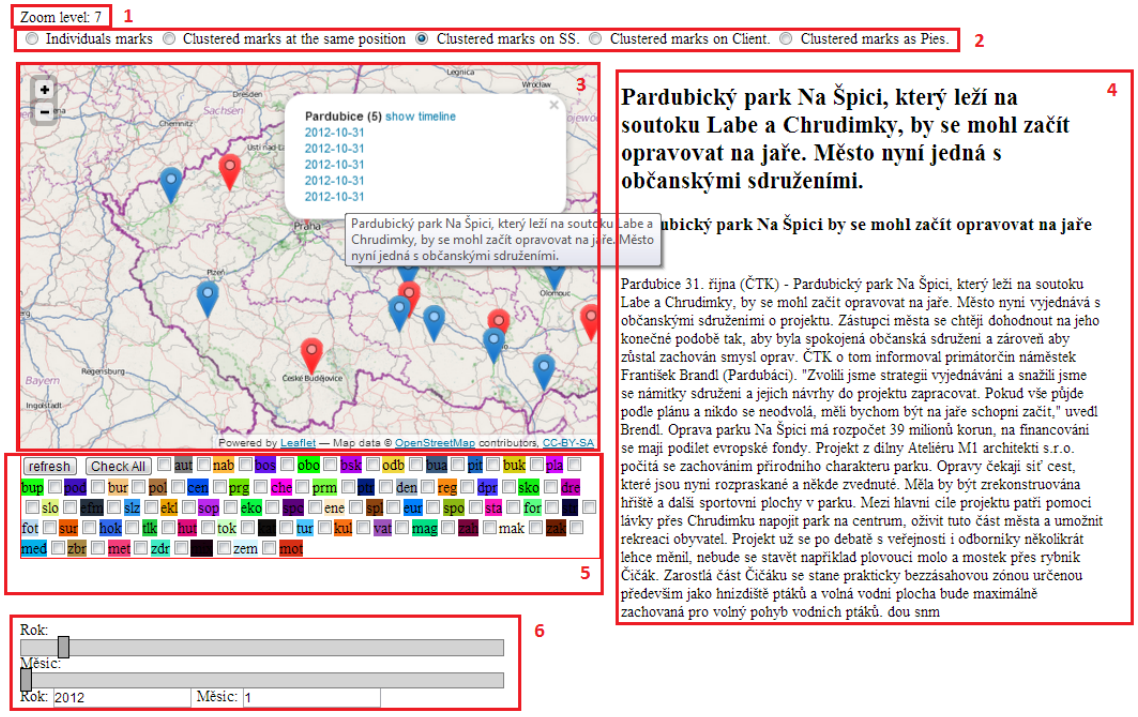


Obrázek 6.4: Vývojový diagram algoritmu pro výpočet centroidu shluků

- **3** - Mapa, místo pro geografické rozmístění značek (*DIV* element s *id="map"*).
- **4** - Oblast pro zobrazení obsahu značek na mapě (*DIV* element s *id="newsItem"*). Zobrazovaným obsahem mohou být textová data konkrétní značky, nebo časová osa, která je naplněna daty z daných GPS souřadnic (odkaz *shown timeline* ve vyskakovacím okně)[5].
- **5** - Formulář s *id="categories"* pro práci s kategoriemi. Kategorie se využívají při zobrazení dat koláčovými grafy (volba *"clustered marks as Pies"* ve formuláři označeném číslem 2), nebo pro filtrování dat na časové ose, která může být zobrazena v oblasti s číslem 4.
- **6** - Posuvníky filtrující zobrazovaná data na mapě na základě nastaveného času[6]. Zobrazena jsou všechna data, která mají shodný rok s rokem vybraným na posuvníku určujícím rok, a která v datumu obsahují měsíc předcházející nebo shodný měsíc s vybraným na posuvníku určujícím měsíc.

[5]Zobrazování dat současně i na časové ose není implementováno pro značky reprezentující shluk vytvořený algoritmem popsaným v sekci 6.3.7 (na mapě jsou takovéto značky odlišeny zelenou barvou)

[6]Filtrování implementováno pouze pro "koláčové zobrazení dat" (volba *"clustered marks as Pies"* ve formuláři označeném číslem 2)



Obrázek 6.5: Klientská část aplikace v API Leaflet

Po načtení stránky s aplikací jsou zobrazena všechna data tak, že jsou kompletně nashlukována podle algoritmu z kapitoly 6.3.7 a následně je pro každou samostatnou značku či shluk vytvořena jedna značka.

Funkčnost aplikace se převážně odvíjí od vybrané volby zobrazení. Obecně platí, že u všech voleb jsou na mapu umístěny značky, které mohou být ruzných typů.

Jednotlivé typy značek jsou od sebe barevně odlišeny. Modrá barva reprezentuje individuální značky, které obsahují data o jediné položce. Červenou barvou jsou tvořeny značky reprezentující více dat na stejných GPS souřadnicích a zelené značky jsou použity pro shluky dat vytvořené na serverové straně. Značky shluků se odlišují také svojí velikostí, která se odvíjí od počtu dat, ze kterých shluk vznikl. Shluky obsahující méně než 20 značek mají stejnou velikost jako jednotlivé značky.

S každou značkou je spjaté vyskakovací okno, jehož obsah se liší podle typu (barvy) značky. Většinou obsahuje název města, ve kterém je značka umístěna, případně počet jednotlivých datových záznamů na dané pozici a odkaz na každý z nich. Obsah jednotlivých záznamů se po kliknutí na odkaz zobrazí v pravé části obrazovky. Pokud se nejedná o zelené značky, obsahuje záznam také odkaz na časovou osu, která se zobrazí v pravé části obrazovky.

Módy zobrazení

Jednotlivé módy zobrazení odpovídají položkám formuláře s číslem 2) na obrázku 6.5.

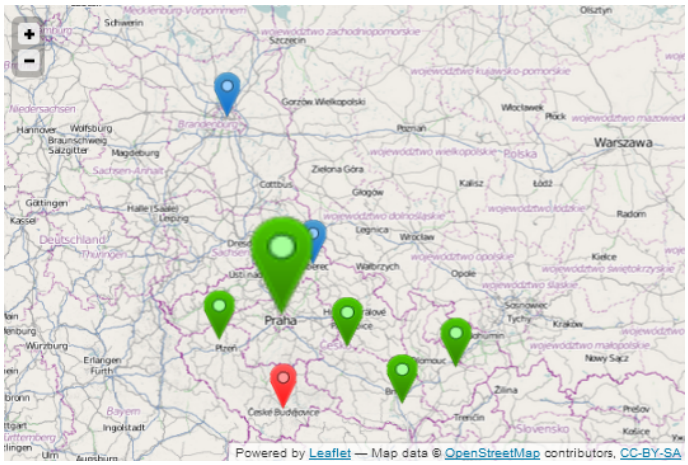
- *Individual marks* - Základní rozmístění značek pomocí API. Vyskakovací okna obsahují datové záznamy na dané pozici, které je možné zobrazit. Všechna data ze serveru jsou předána formou seznamu a pro každou položku tohoto seznamu je vytvořena samostatná značka.

- *Clustered marks at the same position* - Zredukování dat se stejnými GPS souřadnicemi na serverové straně. Značky obsahující více datových záznamů jsou odlišeny červenou barvou. Všechny značky obsahují vyskakovací okna s jednotlivými záznamy, které je možné zobrazit a odkaz pro zobrazení časové osy naplněné daty reprezentovanými danou značkou. Data ze serveru jsou předána v podobě hašovací tabulky, kde klíčem jsou GPS souřadnice a hodnotou je vždy jeden záznam obsahující metadata o datech na těchto souřadnicích. Tyto metadata obsahují např. název města, datum a čítač, který určuje počet dat na daných souřadnicích.

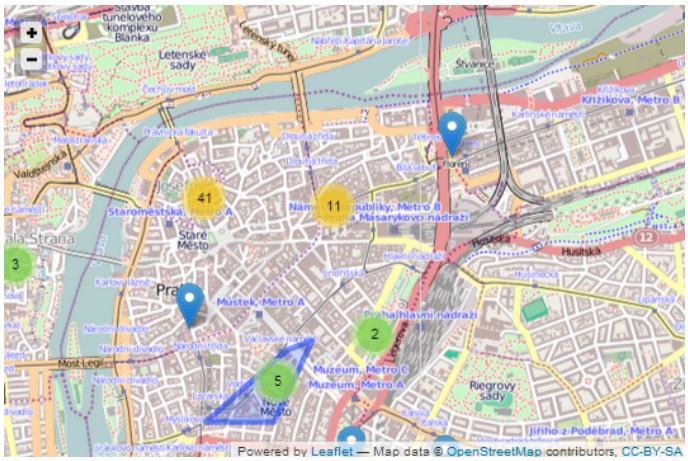
- *Clustered marks on SS* - Značky jsou nashlukovány na serverové straně podle algoritmu viz 6.3.7. Značky které takto vznikly uvedeným shlukovacím algoritmem na serveru jsou odlišeny zelenou barvou (viz obrázek 6.6) a mohou nabývat různých rozměrů v závislosti na množství dat ve shluku. Při změně měřítka se odesílá na server nový požadavek a shluky jsou znovu přepočítány. Pokud chceme zobrazit obsah shluku, je nutné zoom přiblížit tak, aby se shluk (zelená značka) rozdělil na červené a modré značky. Pro snížení přenášeného objemu dat totiž tyto shluky neobsahují data, podle z nichž by bylo možné identifikovat konkrétní položky, ze kterých jsou složené.

- *Clustered marks on client* - Shluky vytvořené na straně klienta použitím pluginu *Leaflet.markercluster*. Ukázka je vidět na obrázku 6.7. Při najetí kurzoru myši na shluk je zobrazena hranice oblasti, ze které je shluk tvořen. Ze serveru jsou totiž posílány jednotlivé záznamy (včetně GPS souřadnic) a shlukování probíhá na klientské straně. Výhodou tohoto řešení je pouze jeden datový požadavek na server, nicméně nevýhodou je, že rozmístění značek na mapě je řízeno pluginem a poté se se značkami nedá nijak manipulovat.

- *Clustered marks as Pies* - Zobrazení shluků (rozmístění dat získávaných ze serveru je totožné jako u volby *Clustered marks on SS*) formou koláčových grafů, které zobrazují procentuální podíl kategorií v dané oblasti (viz obrázek 6.8). Dané kategorie, které chceme v grafech zobrazit se vyberou ve formuláři umístěném pod mapou. Poté stačí stisknout tlačítko *refresh*. Procentuální podíl daných kategorií je možné zobrazit ve vyskakovacím okně kliknutím na daný koláč. U této volby zobrazení je také jako u jediné implementován vývoj v čase, kdy lze měnit rok a měsíc na k tomu určených posuvnících a pozorovat změny v koláčových grafech kategorií na mapě.



Obrázek 6.6: Zobrazení shluků vytvořených na serverové straně aplikace



Obrázek 6.7: Zobrazení shluků vytvořených pluginem Leaflet.markercluster

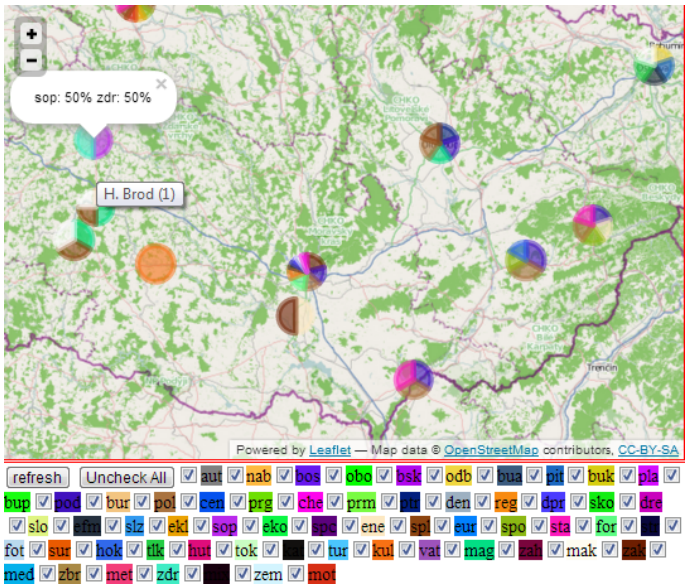
6.4.2 Použité knihovny

Vytvořená aplikace využívá knihovny: *Leaflet*, *Almende Timeline*, *JQuery*, *JAK* a *slider.js*[7].

Dále aplikace používá následující pluginy pro *Leaflet API*:

- *leaflet.label* - plugin pro tvorbu popisků
- *leaflet.markercluster* - plugin pro shlukování značek
- *leaflet.semicircle* - plugin rozšiřující vlastnosti kruhu (kreslení vektorové grafiky)

[7] widget pro vytvoření posuvníku http://jak.seznam.cz/example/widgets/#slider



Obrázek 6.8: Zobrazení shluků formou koláčových grafů

Pomocí elementu *<script type="text/javascript"src="cesta ke knihovne"/>*, který je v hlavičce html stránky s aplikací, se připojují všechny skripty, knihovny a pluginy. Seznam všech dostupných pluginů pro *API Leaflet* je dostupný na adrese http://leafletjs.com/plugins.html.

6.4.3 Struktura

V této podkapitole je nejprve popsána adresářová struktura aplikace. Následuje popis konfiguračního skriptu aplikace a HTML stránek.

Adresářová struktura a popis souborů

Aplikace je umístěna ve složce *WebApplicationCTK/leaflet*, která obsahuje složky *css, img a scripts*. Zde jsou obsažené definice kaskádových stylů (css), obrázky reprezentující značky umisťované na mapě a zdrojové kódy aplikace - skripty. Dále jsou v této složce stránky *LEAFLET.html* a *googleleaflet.html*, která demonstruje použití API s mapami od společnosti Google.

Ve složce *leaflet/scripts* jsou umístěny tyto soubory:

- *leaflet_label* - adresář se zdrojovými kódy pluginu *leaflet.label*
- *leaflet_marker_cluster* - adresář se zdrojovými kódy pluginu *leaflet.markercluster*
- *leaflet_semicircle* - adresář se zdrojovými kódy pluginu *leaflet.semicircle*
- *timeline* - adresář se zdrojovými kódy knihovny *Almende Timeline*

- *config.js* - konfigurace aplikace
- *ctk_timeline.js* - script s definicí časové osy
- *markers.js* - skript s definicí funkcí umisťujicích značky na mapu pro různé módy
- *timeline_utils.js* - pomocné funkce pro práci s kategoriemi v časové ose
- *utils.js* - pomocné funkce celé aplikace

config.js

Konfigurační soubor, ve kterém jsou definovány následující konstanty pro nastavení aplikace:

- *SERVER_URL, TIMELINE_SERVER_URL* - řetězce udávající URL adresu serverové části aplikace
  (defaultní hodnota: *http://localhost:8084/WebApplicationCTK/*)
- *CATEGORIES, TIMELINE_CATEGORIES* - pole s názvy kategorií ČTK
- *TIMELINE_RANGE* - rozsah zobrazení periodických událostí, periodické události budou vyobrazeny v tomto rozsahu před a po současném datu (defaultní hodnota: *50*)
- *TIMELINE_SOURCE_URL* - URL pro získání dat (defaultní hodnota: *TIMELINE_SERVER_URL + "/GetData?show=timeline"*)

LEAFLET.html

LEAFLET.html je hlavní stránka pro přístup k aplikaci. V hlavičce stránky jsou vloženy všechny použité knihovny a skripty. Tělo stránky obsahuje ovládací elementy popsané v kapitole 6.4.1 a zobrazené na obrázku 6.5.

Za těmito elementy následuje definice skriptu, který inicializuje celou aplikaci. Vytvoří mapu, vrstvy pro značky, listenery událostí, vygeneruje checkboxy pro kategorie a časové posuvníky spolu s jejich listenery. Aplikace používá mapy *OpenStreetMap*.

googleleaflet.html

Html stránka, která demonstruje použití API *Leaflet* s mapovými podklady od společnosti Google. Kombinaci *Google Maps a Leaflet* řeší plugin *leaflet-google* [8]. V hlavičce html stránky musí být vloženo elementem *<script>* API *Leaflet* (+ CSS styly), API *Google Maps* a knihovna *leaflet-google* viz následující kód:

[8] https://gist.github.com/bencevans/4504864/raw/c9ef880071f959398b7cf0b687d4f37c352ea86d/leaflet-google.js

```
<link rel="stylesheet" href="http://cdn.leafletjs.com/
    leaflet-0.4.5/leaflet.css" />
<script src="http://cdn.leafletjs.com/leaflet-0.4.5/
    leaflet.js"></script>
<script src="http://maps.google.com/maps/api/js?v=3.2&
    sensor=false"></script>
<script src="https://raw.github.com/gist/2197042/2
    b90c41b39b7d5b3a851d8f256de2ebd3fe1fb74/leaflet-
    google.js"></script>
```

Vytvoření mapy používající podklady od Google se poté provede v javascriptu následovně:[9]

```
var map = new L.Map('map', {center: new L.LatLng
    (51.51, -0.11), zoom: 9});
var googleLayer = new L.Google('ROADMAP');
map.addLayer(googleLayer);
```

6.5 Klientská část - Almende Timeline

Tato kapitola popisuje druhou klientskou část aplikace vytvořenou pomocí nástroje *Almende Timeline* (viz 5.7), která slouží pro vizualizaci dat na základě časových údajů formou časové osy. Ke své správné funkci nepotřebuje přístup k internetu.

6.5.1 Popis funkčnosti

Stránka zobrazující časovou osu je v aplikaci dostupná na adrese *WebApplicationCT-K/timeline/timeline.html*.

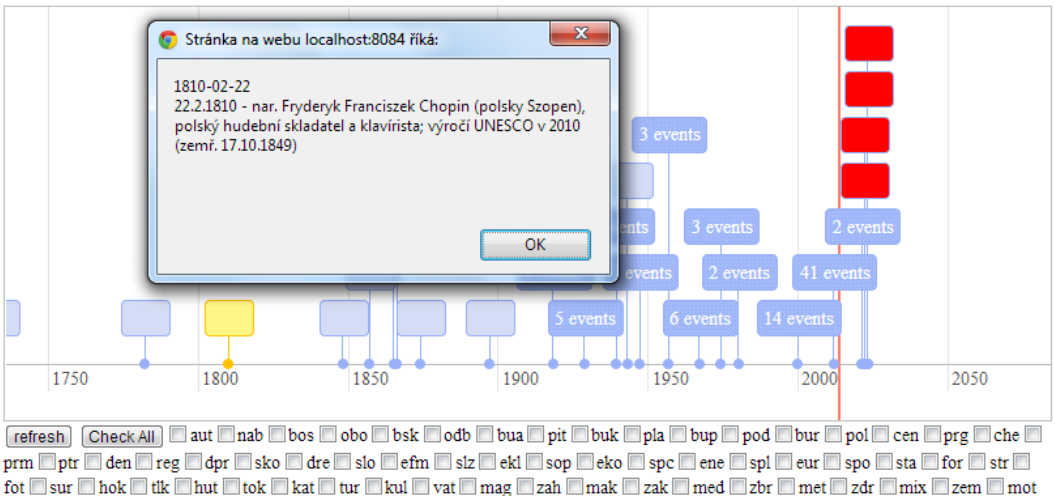
Zobrazovaná data se získávají ze serveru (viz 6.3.2) na základě požadavku, který je na server odesílán během načítání stránky s aplikací.

Po načtení stránky jsou všechna data ze serveru zobrazena jako samostatné buňky přichycené k časové ose. Obsah buňek lze zobrazit najetím kurzoru myši nebo kliknutím na buňku (viz obrázek 6.9). Periodické události jsou odlišeny červenou barvou. Časovou osu lze libovolně posouvat v čase tahem kurzoru myši a zvětšovat či zmenšovat zobrazovanou část rolovacím kolečkem myši (zoom). Při změně zoomu se zobrazované měřítko na časové ose samo přepočítá. Měřítko časové osy se dá měnit v rozsahu roků až milisekund.

Aplikace využívá shlukování událostí, které API nabízí. Události se stejným nebo blízkým časem jsou seskupeny do jedné buňky zobrazující počet seskupených událostí (viz obrázek 6.9).

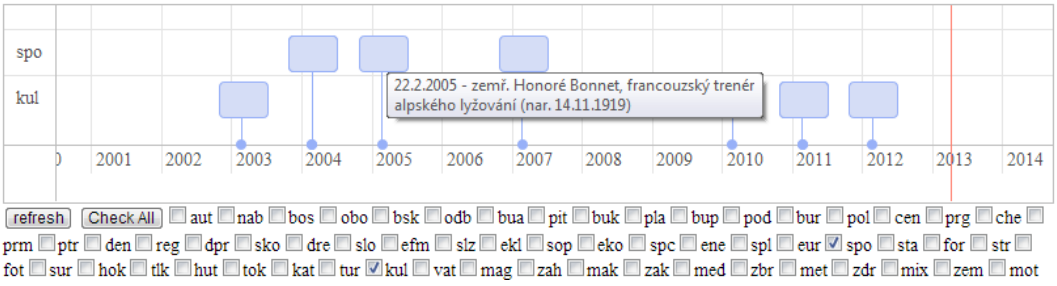
Aplikace dále nabízí filtrování zobrazovaných událostí podle kategorií. K tomu slouží formulář umístěný pod časovou osou, ve kterém jsou umístěny checkboxy

[9] Za předpokladu, že je na stránce DIV element s id="map"(první parametr u L.Map())



Obrázek 6.9: Ukázka časové osy

pro každou kategorii. Kategorie jsou definovány v konfiguračním skriptu aplikace. Po zaškrtnutí vybraných kategorií a stisku tlačítka *refresh* se na časové ose zobrazí pouze událostí z daných kategorií (pokud data takové události obsahují). Pro každou ze zaškrtnutých kategorií se v časové ose vytvoří vrstva, ve které jsou zobrazeny pouze události z dané kategorie (viz obrázek 6.10). Vertikální rozměr časové osy se mění v závislosti na počtu vytvořených vrstev.



Obrázek 6.10: Filtrování - časová osa s vrstvami pro události z kategorií kultura a sport

6.5.2 Použité knihovny

V aplikaci musí být vloženy všechny skripty popsané v následující kapitole, knihovna *Almende Timeline* a knihovna *JQuery*. Všechny skripty a knihovny se připojují elementem *<script type="text/javascript"src="cesta ke knihovne"/>*, který je v hlavičce html stránky s aplikací.

6.5.3 Struktura

Na začátku této podkapitoly je nejdříve popsána adresářová strukura aplikace. Následuje popis klíčových skriptů aplikace, včetně detailnějšího popisu významných funkcí a použitých javascriptových konstrukcí.

Adresářová struktura

Aplikace je umístěna ve složce *WebApplicationCTK/timeline/*, která obsahuje složku *scripts* a stránku *timeline.html* (viz 6.5.3) se samotnou aplikací.

Ve složkce *timeline/scripts* jsou umístěny tyto soubory:

- *config.js* - konfigurace aplikace
- *ctk_timeline.js* - script s definicí časové osy
- *timeline_utils.js* - pomocné funkce pro práci s kategoriemi
- *timeline* - adresář se zdrojovými kódy knihovny *Almende Timeline*

config.js

Konfigurační soubor ve kterém jsou definovány následující konstanty pro nastavení aplikace:

- *TIMELINE_SERVER_URL* - řetězec udávající URL adresu serverové části aplikace (defaultní hodnota: *http://localhost:8084/WebApplicationCTK/*)
- *TIMELINE_RANGE* - rozsah zobrazení periodických událostí, periodické události budou vyobrazeny v tomto rozsahu před a po současném datu (defaultní hodnota: *10*)
- *TIMELINE_SOURCE_URL* - URL pro získání dat (defaultní hodnota: *http://localhost:8084/WebApplicationCTK/GetData?show=timeline"*)
- *TIMELINE_CATEGORIES* - pole s názvy kategorií ČTK

ctk_timeline.js

Skript s definicí funkce pro vizualizaci dat ČTK pomocí API *Almende Timeline*. Obsahuje funkci *timeline_drawVisualization(timelineId, sourceUrl, checkedCategories)*. Popis parametrů funkce:

- *timelineId* - id *DIV* elementu pro časovou osu
- *sourceUrl* - URL pro získání dat
- *checkedCategories* - pole obsahující názvy zaškrtnutých kategorií

Data pro vizualizaci jsou přijímána v JSON formátu a získávají se pomocí Ajaxu dotazem na server. Následující části kódu demonstrují základní princip vytváření časové osy, tj. inicializaci a vložení dat. Detaily které zohledňují rozdílné typy dat (viz kapitola 2), práci s kategoriemi, vrstvami a periodickými událostmi nebo vytváření listenerů jsou podrobně komentované ve zdrojovém kódu.

1) Deklarace:

```
var timelineData = [];//pole pro data
var timeline;// časová osa
var options = {...};//nastavení časové osy
```

2) Přidání události do časové osy, kde jednotlivá data mohou mít strukturu JSON nebo Google DataTable:

```
timelineData.push({
  'start': new Date(date), //čas události
  'content': "<div style=\"width:30px;\" title=\"" +
     title + "\"> </div>",//vzhled a obsah buňky
  'jsonData': data[i] //další data ve formátu JSON
});
```

3) Vytvoření objektu časové osy a vykreslení časové osy:

```
timeline =
new links.Timeline(document.getElementById(timelineId));
timeline.draw(timelineData, options);
```

Standardní událost má 2 povinné a 4 volitelné parametry[10]. Díky tomu, že struktura události používá JSON formát, je možné přidat i další parametry k události (viz parametr *jsonData* v bodě 2) ve výše uvedeném kódu). K takovýmto parametrům není možno přistupovat přímo pomocí API, ale lze tato data získat následujícím způsobem:

```
var data = timeline.getData()[row].jsonData;
```

Tímto způsobem lze k události připojit libovolná data.

timeline.html

Html stránka pro přístup k aplikaci. V elementu *BODY* volá při načtení funkci *timeline_drawVisualization('timeline', TIMELINE_SOURCE_URL)* ze skriptu *ctk_timeline.js*. Ve svém těle obsahuje pouze jeden *DIV* element, který slouží jako kontejner pro časovou osu a definici formuláře (jednotlivé prvky pro kategorie se generují dynamicky).

Obsah elementu *BODY*:

[10] Viz dokumentace
http://almende.github.com/chap-links-library/js/timeline/doc/#Data_Format

```
<div id="timeline"></div>
<form id="categories" name="categories">
  <input type="button" value="refresh" onClick="
      timeline_refreshVizualization('timeline',
      TIMELINE_SOURCE_URL);">
</form>
```

Za tímto kódem následuje skript, který volá funkci pro vygenerování formulářových prvků jednotlivých kategorií ze souboru *timeline_utils.js* a definuje funkci pro aktualizaci časové osy při změně vybraných kategorií (volá funkci *timeline_drawVizualization(timelineId, sourceUrl, checkedCategories)* se všemi jejími parametry). Vybrané kategorie, které potřebujeme jako třetí parametr této funkce dostaneme zavoláním následující funkce definované v souboru *timeline_utils.js*:

```
timeline_getCheckedValues(document.forms['categories'].
    elements['category'])
```

Kapitola 7

Dosažené výsledky

7.1 Testování shlukovacího algoritmu

Vzhledem k tomu, že jsem dat s GPS souřadnicemi měl od ČTK pouze omezené množství, vygeneroval jsem si pro tyto účely náhodně 50 000 unikátních GPS souřadnic z okolí Prahy, jejichž maximální vzdálenost od Prahy činila 300 km. K tomu jsem využil *Random Point Generator*[1].

Výpočetní složitost shlukovacího algoritmu nelze přesně určit, protože velmi záleží na samotných datech, ne jen na jejich počtu. Asymptotickou složitost algoritmu lze omezit zhora i zdola. V ideálním případě (dolní mez) je složitost algoritmu lineární *O(N)*. Pro nejhorší případ je složitost shora omezena kvadratickou funkcí a spadá do třídy $O(N^2)$. Algoritmus má lineární složitost v případě, že všechna data reprezentovaná GPS souřadnicemi se nachází „blízko“ sebe[2]. V opačném případě, kdy jsou všechna data od sebe „příliš“ vzdálena (může být způsobeno např. velkým přiblížením - zoomem) je časová složitost algoritmu téměř kvadratická a nemusí být vytvořen vytvořen žádný shluk.

Výsledky testování časové náročnosti algoritmu jsou uvedeny v tabulce 7.1.1 a zobrazeny v grafu 7.1. U *OpenStreetMaps*, které jsem použil jako mapové podklady se zoom skládá z osmnácti úrovní (1-18), přičemž hodnota 18 je největší možné přiblížení. Tabulka 7.1.1 obsahuje pouze úrovně 1-6, přičemž poslední řádek[3] ukazuje teoreticky nejhorší možný případ z hlediska asymptotické složitosti, kdy výstup algoritmu je roven jeho vstupu.

[1] http://www.geomidpoint.com/random/

[2] Vzdálenost dat se zde odvíjí od úrovně zoomu, podle kterého je definována mez určující zda jsou data „blízko“ sebe a mohou tvočit shluk či nikoli

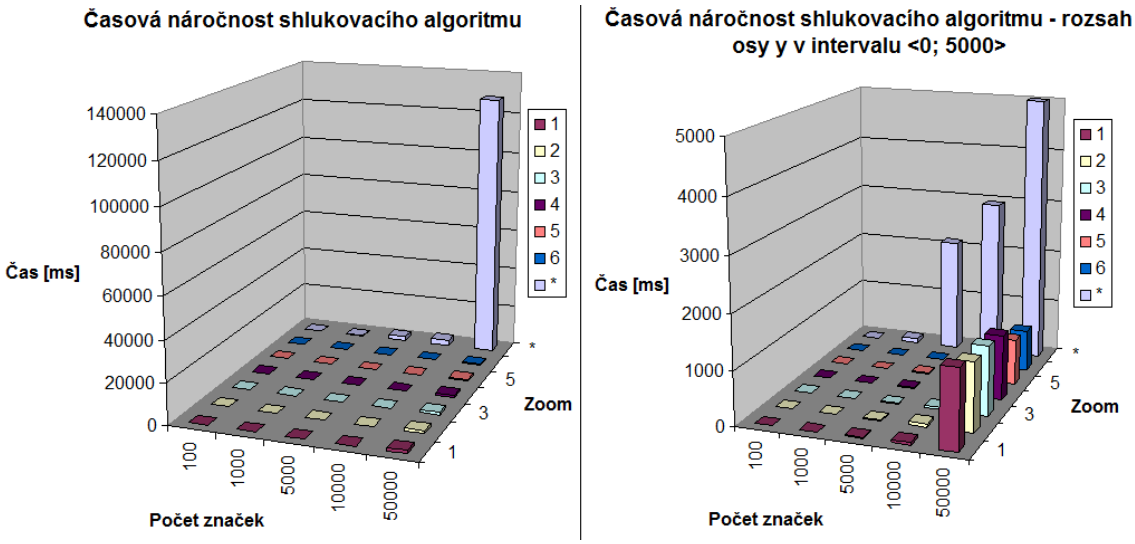
[3] Tyto hodnoty byly zjištěny během testování se špatně implementovanou funkcí pro výpočet minimální vzdálenosti, kdy pro úroveň zoomu větší než 6 algoritmus přestal vytvářet shluky. V současné době je tento problém již odstraněn a algoritmus pracuje správně pro všechny úrovně zoomu.

7.1.1 Konfigurace počítače

Testy shlukovacího algoritmu byly prováděny na počítači s následující softwarovou a hardwarovou konfigurací: 1) SW: Windows 7 Professional, Java JDK 1.6, Apache Tomcat 6.0.18; 2) HW: Intel Core 2 Duo T5870, 4 GB RAM.

Počet značek	100			1000			5000			10000			50000		
Zoom	t_1	t_2	P	t_1	t_2	P	t_1	t_2	P	t_1	t_2	P	t_1	t_2	P
1	0	0	1	2	0	1	20	0	1	57	3	1	1455	15	1
2	0	0	1	2	0	1	18	0	1	58	3	1	1278	11	1
3	0	0	1	2	0	1	15	0	1	57	1	1	1286	4	1
4	0	0	2	2	0	4	12	0	3	48	1	3	1201	5	3
5	0	0	5	2	0	7	12	0	7	40	0	8	856	6	9
6	0	0	22	3	0	38	20	1	43	56	1	44	722	8	61
*	1	0	0	80	0	0	2126	0	0	2955	0	0	128513	0	0

Tabulka 7.1: Časová náročnost shlukovacího algoritmu. P představuje počet vzniklých shluků, t_1 dobu vytvoření shluků [ms], t_2 dobu výpočtu centroidů [ms]



Obrázek 7.1: Časová náročnost shlukovacího algoritmu v závislosti na počtu značek a úrovni zoomu

7.1.2 Shrnutí

Jak je patrné z tabulky 7.1.1 a grafu 7.1 shlukovací algoritmus s vyjímkou nejhoršího případu kdy nevznikne žádný shluk dosahuje velice dobrých výsledků i přes to, že jsem v testu použil data s unikátními GPS souřadnicemi. Časová náročnost výpočtu centroidů je ve všech případech zanedbatelná (v řádech milisekund). Na reálných datech ČTK bude probíhat proces shlukování mnohem rychleji, protože na daném vzorku dat má pouze 20% dat unikátní GPS souřadnice. Tuto vlastnost dat je možno využít pro předzpracování a zredukovat tak vstup algoritmu pouze na unikátní souřadnice. Toho jsem v aplikaci docílil použitím hašovací tabulky, jejíž klíč tvořily GPS souřadnice a hodnotu *MarkBean*. Stejné GPS souřadnice nevložily do hašovací tabulky nový prvek, pouze v již existujícím záznamu inkrementovaly čítač reprezentující počet značek na těchto souřadnicích.

Nevýhoda použitého shlukovacího algoritmu spočívá v případech, kdy jsou od sebe data příliš vzdálena a nezvniká žádný shluk (viz tabulka 7.1.1), což při použití OpenStreetMap nastává zpravidla při velkém přiblížení. Provedení algoritmu pak zabere hodně času s minimálním přínosem.

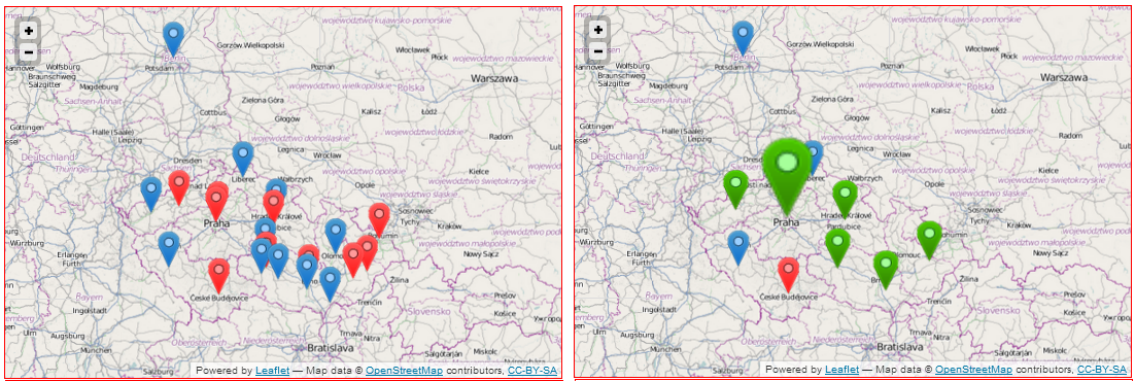
K odstranění tohoto problému bych navrhoval před shlukováním omezit množinu dat pouze na data s GPS souřadnicemi v aktuálním výřezu mapy. *API Leaflet* obsahuje metodu pro získání GPS souřadnic hranic aktuálního výřezu mapy[4]. Doporučil bych však zvážit, zda se na data z výřezu nedotazovat pouze od určité úrovně zoomu či velikosti dat. Důvodem je možné posouvání mapy, při kterém se by se v takovém případě musela data ze serveru načítat znovu při každém posunutí.

7.2 Geografická vizualizace

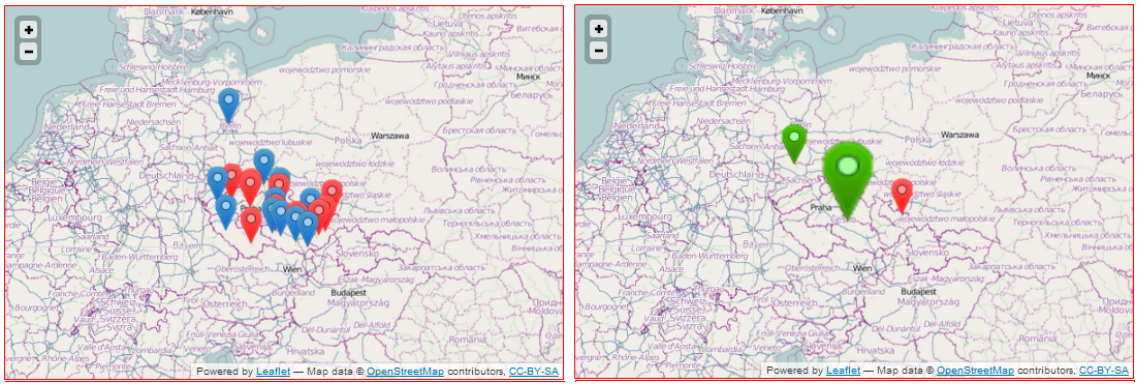
Hlavní problém u geografické vizualizace bylo přehledné zobrazení většího počtu značek. Toho jsem docílil předzpracováním dat se stejnými GPS souřadnicemi a shlukováním, které jsem implementoval na serverové straně aplikace. Shlukovací algoritmus vytváří shluky podle vzájemné vzdálenosti dat a úrovně zoomu. Výsledky implementovaných postupů jsou vidět na obrázcích 7.2, 7.3 a 7.4. Modré značky zde reprezentují jediný záznam, červené značky více záznamů na stejných GPS souřadnicích a zelenou barvou jsou označeny shluky vytvořené pomocí shlukovacího algoritmu. Na obrázku 7.5 je vidět kombinace shlukování značek a jejich následná reprezentace koláčovými grafy, které reprezentují procentuální zastoupení kategorií dat na daných GPS souřadnicích. Konkrétně se jedná o zobrazení kategorií *Politika (pod), Politika ČR (pol), Parlamenty a vlády (for).*

Cílem práce bylo prozkoumání možností a způsobů zobrazení geografických dat, nikoli vytvoření konkrétní aplikace. Hlavní požadavky ze strany zadavatele byly následující: 1) vhodné API; 2) celosvětové mapové pokrytí; 3) shlukování značek; 4) zobrazení kategorií; 5) vývoj na mapě z hlediska času. Vytvořená aplikace všechny tyto požadavky splňuje. Vybráno bylo *API Leaflet* v kombinaci s *OpenStreetMap*,

[4]Viz http://leafletjs.com/reference.html#map-get-methods



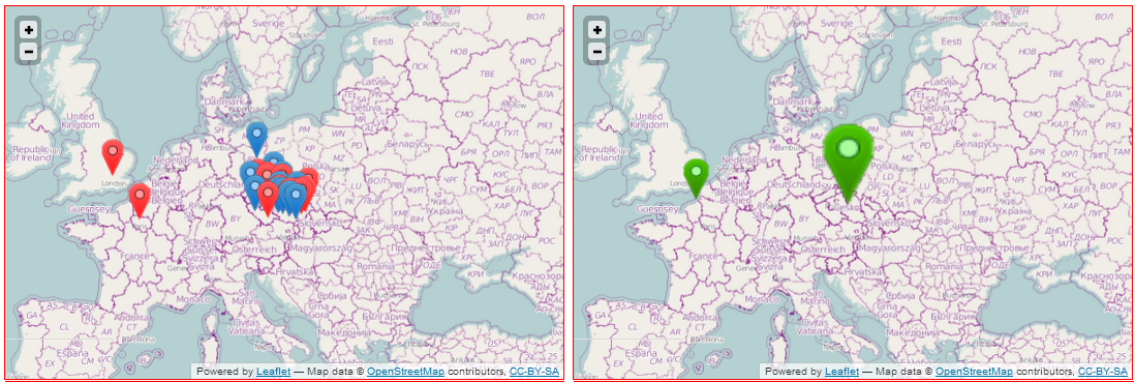
Obrázek 7.2: Geografická vizualizace (zoom = 6) - neshlukované značky (vlevo) a shlukované značky (vpravo)



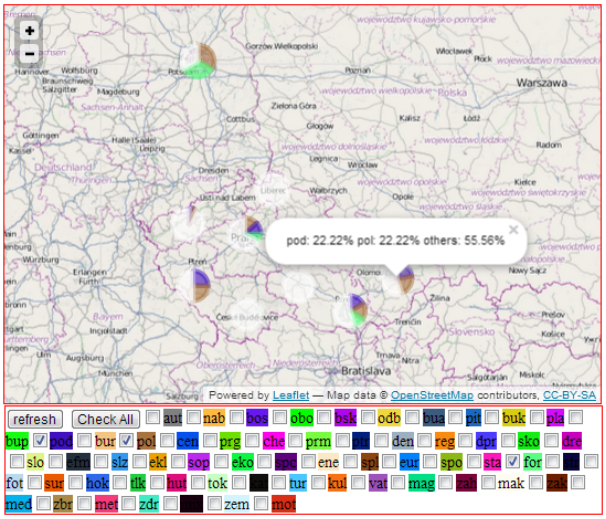
Obrázek 7.3: Geografická vizualizace (zoom = 5) - neshlukované značky (vlevo) a shlukované značky (vpravo)

které zajišťují globální mapové pokrytí. Shlukování značek je realizováno implementovaným shlukovacím algoritmem na serverové straně a také pluginem pro API Leaflet na klientské straně. Místo značek je možné zobrazit na mapě vybrané kategorie formou koláčových grafů, které reprezentují procentuální podíl dat z kategorií v dané oblasti. Jejich vývoj lze sledovat v čase posouváním příslušných ovládacích prvků (posuvníků).

Implementaci posuvníku určujícího rok by pro budoucí využití bylo vhodné upravit. V současné implementaci se tahem posuvníku myší generuje (pro každou změnu hodnoty posuvníku) požadavek na server. Lze tak docílit plynulého vývoje v čase, ale za cenu zvýšení zátěže na serveru. Pokud uživatel bude chtít např. zobrazit konkrétní rok vzdálený o x let od současné hodnoty a přesune posuvník na tento rok tahem, vygeneruje se zbytečně x požadavků. Pro zamezení těchto situací bych doporučoval obsluhu posuvníku určující rok zaregistrovat na jinou událost (např.



Obrázek 7.4: Geografická vizualizace (zoom = 4) - neshlukované značky (vlevo) a shlukované značky (vpravo)



Obrázek 7.5: Geografická vizualizace - zobrazení koláčových grafů

puštění posuvníku tlačítkem myši) a pro zachování vývoje v čase bych k posuvníku přidal tlačítka posouvající posuvník o „+/-“ 1 rok.

7.3 Vizualizace časových dat

V aplikaci zobrazující data na časové řadě jsem nemusel řešit žádné problémy. Knihovna *Almende Timeline* 5.7, kterou jsem pro realizaci této aplikace použil je velmi dobře navržená a umožňuje automatické shlukování většího počtu značek na časové ose. Shlukování funguje obdobně jako implementované řešení pro geografickou vizualizaci popsané v této práci. Podle úrovně zoomu a vzdálenosti dat (velikost

časového intervalu) značky seskupuje do shluků.

Aplikace komunikuje se serverem pouze v případě načtení, nebo při použití filtru vybraných kategorií. Na serverové straně neprobíhá žádné shlukování, pouze parsování dat z XML formátu. Proto bych pro vylepšení této aplikace navrhoval zaslat rozparsovaná data najednou tak jako je to doposud a realizovat filtrování kategorí v JavaScriptu aby se odstranila zbytečná zátěž na serveru. Na rozdíl od mapových API dokáže knihovna *Almende Timeline* zobrazovat na časové ose plynule 10 000 záznamů [14].

Knihovna *Almende Timeline* se neustále vyvýjí a doporučuji sledovat její webové stránky, neboť mezi vytvořením aplikace využívající tuto knihovnu a sepsáním tohoto textu vyšly dvě nové verze, které přinášejí nové funkce.

Kapitola 8

Závěr

V počáteční fázi této práce jsem se věnoval analýze dat České tiskové kanceláře a vhodných volně dostupných nástrojů, které by bylo možné využít pro vizualizaci dat a to zejména pro komerční použití. Výsledkem této fáze je přehled sedmi nástrojů společně se srovnáním jejich výhod a nevýhod.

Na základě této analýzy jsem vybral dva nástroje. *API Leaflet* pro geografickou vizualizaci dat a knihovnu *Almende Timeline* pro zobrazení dat z pohledu času. Pomocí těchto nástrojů jsem implementoval dvě webové aplikace klient-server, které různými způsoby vizualizují data ČTK typu *události* a *dokumenty*.

První aplikace zobrazuje data podle jejich pozice uvedené v metadatech. Umožňuje zobrazení dat na mapě z různých pohledů včetně koláčových grafů (na základě kategorií) a jejich vývoje z pohledu času. Uspokojivě řeší problém zobrazení většího množství dat využitím shlukovacího algoritmu. Shlukovací algoritmus je vlastní modifikací existujícího algoritmu [2]. Algoritmus shlukuje data na základě jejich vzdálenosti a úrovně zoomu. Jeho předností je rychlost shlukování dat a kvalitní reprezentace shluků.

Druhá samostatná aplikace zobrazuje data na časové ose. Problém zobrazení většího množství dat v této aplikaci řeší použitá knihovna *Almende Timeline* na klientské straně, která umožňuje shlukování dat. Události na časové ose lze filtrovat podle kategorií, přičemž pro každou vybranou kategorii je vytvořena nad časovou osou horizontální vrstva obsahující události z dané kategorie. Díky dobrému návrhu je takto implementovanou časovou řadu možno zakomponovat do první aplikace bez nutnosti změn ve zdrojovém kódu.

Přehled zkratek

API Application Programing Interface (aplikační programovací rozhraní)

ČSÚ Český Statistický Úřad

ČTK Česká Tisková Kancelář

DOM Document Object Model (objektový model dokumentu)

EPD Europe's Public Data (portál poskytující veřejné datasety)

GPS Global Positioning System (globální družicový polohový systém)

HTTP Hypertext Transfer Protocol (Komumikační protokol pro výměnu hypermédií)

IPTC International Press Telecommunication Council (mezinárodní organizace vyvíjející technické stanadrdy pro výměnu zpráv)

JAK JAvaScriptová Knihovna

JSON JavaScript Object Notation (textový formát pro výměnu dat)

KPV Key-Value-Pair (dvojice klíč-hodnota)

ODL Open Database Licence (typ softwarové licence)

OGC Open Geospatial Consorium (mezinárodní standardizační organizace pro geografická data)

URL Uniform Resource Locator (jednotný lokátor zdrojů)

WGS 84 World Geodetic System 1948 (světový geodetický standard)

WMS Web Map Service (webová mapová služba)

WMTS Web Map Tile Service (webová mapová dlaždicová služba)

XML Extensible Markup Language (rozšiřitelný značkovací jazyk)

Literatura

[1] OpenGIS® Web Map Tile Service Implementation Standard, 2013. Dostupné z: http://portal.opengeospatial.org/files/?artifact_id=35326.

[2] *Map clustering algorithm* [online]. 2013. [cit. 7. 3. 2013]. Dostupné z: http://stackoverflow.com/questions/1434222/map-clustering-algorithm.

[3] Český statistický úřad. http://www.czso.cz/, 2013.

[4] Europe's Public Data. http://www.publicdata.eu/, 2013.

[5] JQuery: JavaScrippt library. http://jquery.com/, 2013.

[6] JSON: JavaScript Object Notation. https://www.json.org/json-cz.html, 2013.

[7] *Manhattan distance* [online]. 2013. [cit. 2. 5. 2013]. Dostupné z: http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Manhattan_Distance_Metric.htm.

[8] OpenLayers: Free Maps for the Web. http://www.openlayers.org/, 2013.

[9] OpenStreetMap: geographic data for the world. http://www.openstreetmap.org, 2013.

[10] *OpenStreetMap* [online]. 2013. [cit. 7. 3. 2013]. Dostupné z: http://cs.wikipedia.org/wiki/OpenStreetMap.

[11] *The top seven alternatives to the Google Maps API* [online]. 2013. [cit. 8. 3. 2013]. Dostupné z: http://www.netmagazine.com/features/top-seven-alternatives-google-maps-api.

[12] *OpenGIS WMS protocol* [online]. 2013. [cit. 7. 3. 2013]. Dostupné z: http://www.demis.nl/home/pages/wms/docs/opengiswms.htm.

[13] ALLAN, A. Thunderforest maps. http://www.thunderforest.com, 2013.

[14] ALMENDE. Timeline. http://chap.almende.com/timeline, 2013.

[15] CLOUDMADE. Leaflet API. http://leafletjs.com/, 2013.

[16] GOOGLE. Google Maps API. https://developers.google.com/maps/, 2013.

[17] GOOGLE. Gson: A Java library to convert JSON to Java object and vice-versa. https://code.google.com/p/google-gson/, 2013.

[18] MAPY.CZ. Mapy API. http://api4.mapy.cz/, 2013.

[19] MICROSOFT. Bing Maps. http://www.microsoft.com/maps/, 2013.

[20] SEZNAM.CZ. JAK: JavaScriptová Knihovna. https://jak.seznam.cz, 2013.

[21] SIMILE. Timeline. http://www.simile-widgets.org/timeline/, 2013.

[22] ČTK. Formát NewsML ČTK. Technický popis, verze 1.0, 2007.

[23] WWW.IPTC.ORG. IPTC Web - NewsML. http://www.iptc.org/site/News_Exchange_Formats/NewsML_1/, 2013.

[24] WWW.OPENGEOSPATIAL.ORG. OGC Implementation Standards, 2013. Dostupné z: `http://www.opengeospatial.org/standards/is`.

Přílohy

Příloha A

Struktura DVD

Na přiloženém DVD se nachází následující stromová struktura adresářů:

- adresář `doc`
  - adresář `src` – zdrojové LaTeXové soubory této práce
  - adresář `pdf` – tato práce v PDF formátu
- adresář `vizualization` – adresář s webovými aplikacemi a zdrojovými soubory aplikací
  - adresář `javadoc` – adresář s javadoc dokumentací serverové části
  - adresář `lib` – knihovny využívané serverovou částí aplikace
  - adresář `src_klient` – zdrojové soubory klientských částí aplikace
  - adresář `src_server` – zdrojové soubory serverové části aplikace
  - adresář `WebApplicationCTK` – adresář s projektem aplikace pro vývojové prostředí NetBeans
  - `WebApplicationCTK.war` – soubor s vytvořenou webovou aplikací

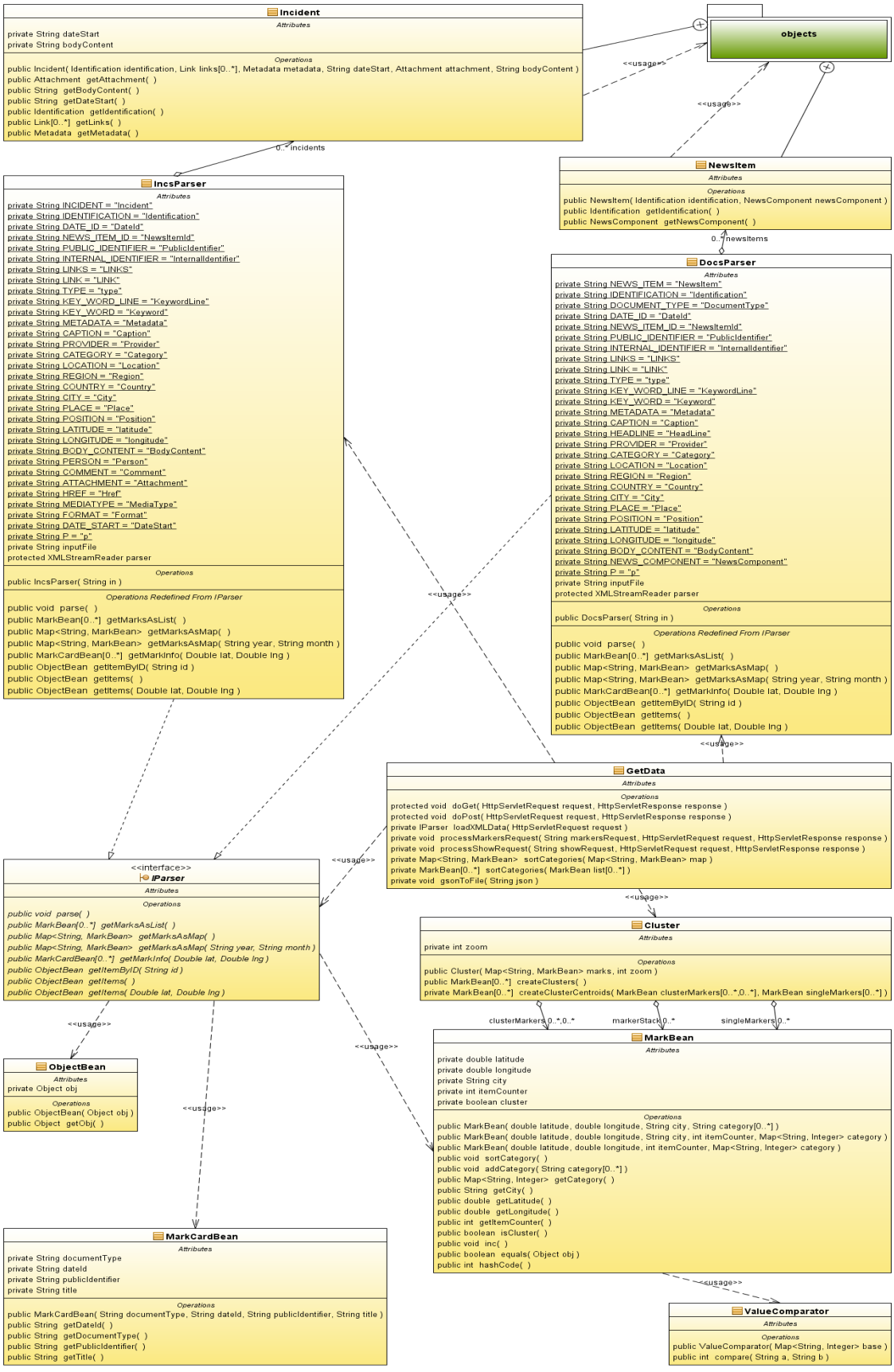
Příloha B

Struktura .war souboru aplikace

- adresář `leaflet` – adresář s klientskou částí aplikace vytvořenou pomocí API Leaflet
- adresář `META-INF` – adresář s manifestem daného .war souboru
- adresář `scripts` – adresář se skripty (knihovnami), které jsou společné pro všechny vytvořené klientské aplikace
- adresář `timeline` – adresář s klientskou částí aplikace vytvořenou pomocí Almende Timeline
- adresář `WEB-INF` – adresář s konfigurací serverové části aplikace a jejími přeloženými třídami
- adresář `index` – úvodní stránka aplikace

Příloha C

UML Diagram serverové části aplikace



Obrázek C.1: UML diagram serverové části aplikace

Příloha D

Uživatelská příručka

Tato kapitola popisuje nahrání aplikace na server (deploy), její nutnou konfiguraci a ovládání. Předpokladem pro správné fungování aplikace je webový server *Apache Tomcat 6.0.18* (dále jen server), pod kterým byla aplikace vyvíjena nebo jeho vyšší verze.

D.1 Deploy aplikace

Aplikace se nachází na přiloženém DVD ve formátu .war souboru. Pro deploy aplikace na server stačí zkopírovat soubor *WebApplicationCTK.war* do adresáře pro webové aplikace, kde se sám rozbalí. Defaultně je tento adresář nastaven na *$catalina_home/webapps*[1]. Definice toho adresáře se nachází v souboru *$catalina_home/conf/server.xml*, konkrétně se jedná o atribut *appBase* elementu *Host*.

D.2 Konfigurace

D.2.1 Serverová část

Konfigurace serverové části se nachází v souboru *WebApplicationCTK/WEB-INF/web.xml*. Serverová aplikace nevyžaduje žádnou dodatečnou konfiguraci pro demonstraci vytvořených aplikací na testovacích datech poskytnutých ČTK.

V souboru *web.xml* je specifikován adresář, obsahující data ČTK pro vizualizaci:

```
<context-param>
  <param-name>document_root</param-name>
  <param-value>data</param-value>
</context-param>
```

Jediná další věc v konfiguraci serveru která stojí za zmínku, je nastavení vhodné délky životnosti *sessions*, aby zbytečně na serveru nezabíraly paměť.

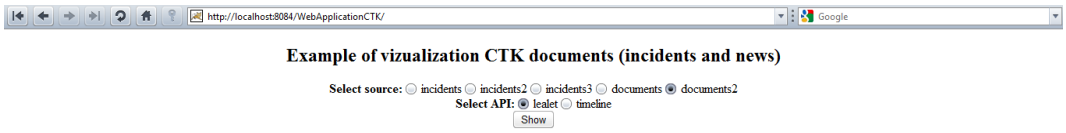
[1] *$catalina_home* - kořenový adresář ve kterém je nainstalovaný Tomcat

D.2.2 Klientské části

Klientské části aplikace se konfigurují souborem v podadresáři *scripts/config.js* daného API. V kapitolách 6.5.3 a 6.4.3 jsou jednotlivé parametry konfigurace detailně popsány.

D.3 Použití aplikace

Pokud je aplikace nahrána na webovém serveru a je spuštěný, stačí do internetového prohlížeče zadat adresu serveru, společně s názvem vytvořené aplikace. Například: *http://localhost:8084/WebApplicationCTK*. Poté se zobrazí úvodní stránka (viz. obrázek D.1, kde je na výběr pět různých dokumentů ČTK a tři API. Na této stránce vybereme API, které chceme pro vizualizaci použít a dokument, který chceme vizualizovat. Tlačítkem *Show* se odešou zvolené parametry na server, který nás poté přesměruje na stránku ve které je implnementovaná vizualizace pomocí vybraného API.



Obrázek D.1: Úvodní stránka aplikace

Popis funkčnosti a ovládání je pro jednotlivá API rozepsán v kapitolách 6.4.1 (pro *API Leaflet*) a 6.5.1 (*Almende Timeline*).

V případě dlouhé nečinnosti, se může stát že na serverové straně vyprší timeout pro session daného uživatele. V takovém případě je při akci na klientské straně, která vyžaduje komunikaci se serverem (např. změna zoomu při vybraném módu, ve kterém probíhá shlukování dat na straně serveru) vypsáno varování a následně je stránka přesměrována zpět na úvodní stránku, kde lze znovu vybrat data pro vizualizaci.