VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÁ APLIKACE PRO EDITACI A ZOBRAZOVÁNÍ HRANIC

WEB APPLICATION FOR EDITATION AND VISUALIZATION OF BORDERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **TOMÁŠ HYPPEŠ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19168/2016/xhypes00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Hypeš Tomáš**
Obor: Informační technologie
Téma: **Webová aplikace pro editaci a zobrazování hranic**
Web Application for Editation and Visualization of Borders
Kategorie: Web

Pokyny:

1. Nastudujte mapy a databázi Openstreet.
2. Vytvořte program pro uložení segmentů hranic mezi obcemi v ČR (případně i okolních států) do databáze Openstreet. Z jednotlivých segmentů vytvořte hranice pro obce a čtvrti v ČR.
3. Navrhněte webovou aplikaci, ve které si registrovaní uživatelé z jednotlivých segmentů (případně celých obcí) v databázi budou moci vytvořit hranice vlastních území (např. panství, farností). K jednotlivým hranicím se budou moci přidávat další údaje (např. roky platnosti hranic).
4. Navrženou aplikaci, včetně správy uživatelů, implementujte.
5. K aplikaci vytvořte návod a dále vytvořte vzorové ukázky hranic pro jednotlivá území.
6. Zhodnoťte použitelnost aplikace a navrhněte další využití.

Literatura:

- Open Street Map, www.openstreetmap.org, 2016

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rozman Jaroslav, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce pojednává o návrhu a tvorbě aplikace pro zobrazování hranic obcí a dalších územních celků a hranic území, která jsou nad těmito územními celky definována. Nejdříve jsou popsány možnosti převodu hranic z databáze OSM a převod z registru RÚIAN. Následně se práce zabývá popisem aplikace pro zobrazování a editaci těchto hranic s možnostmi definovat si vlastní území.

Abstract

This thesis is about design and creation of an application for visualization of village borders and other territorial units and borders of areas, which are defined with them. First, transfer of borders options from OSM database and transfer from register RÚIAN are described. Then the thesis describe the application for visualization and editation this borders with ability to define own areas.

Klíčová slova

Mapy, OSM, OpenStreetMap, RÚIAN, Mapy.cz, hranice, obce, PHP, MySQL, JavaScript

Keywords

Maps, OSM, OpenStreetMap, RÚIAN, Mapy.cz, borders, villages, PHP, MySQL, JavaScript

Citace

HYPEŠ, Tomáš. *Webová aplikace pro editaci a zobrazování hranic*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Webová aplikace pro editaci a zobrazování hranic

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Hyppeš
16. května 2017

Poděkování

Rád bych poděkoval vedoucímu práce, Ing. Jaroslavu Rozmanovi, Ph.D., za konzultace a připomínky při vývoji aplikace.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je vytvořit webovou aplikaci, která bude zobrazovat mapu České republiky. Uživatelé si budou moci v rámci mapy definovat svá vlastní území jako např. panství, farnosti apod. Tato území bude možné vytvářet spojením již existujících území, jako jsou obce, katastrální území, nebo základní sídelní jednotky.

První část této práce pojednává o vytvoření programu pro import těchto existujících území a jejich hranic do vlastní databáze – nejprve z projektu OpenStreetMap (kapitola 2) a následně ze státního registru RÚIAN (kapitola 3). V kapitole 4 je popsána struktura databáze a program, který provádí převod těchto dat do této databáze. Návrh vlastní aplikace se nachází v kapitole 5 a o popisu její realizace pojednávají kapitoly 6 (rozšíření databáze pro účely aplikace), 7 (popis serverové části aplikace), 8 (třídy pro práci s polygony určeny pro operace s hranicemi) a 9 (popis klientské části aplikace). V závěru práce v kapitole 11 je zhodnocen výsledek práce a navrhnuty náměty na další vylepšení.

Kapitola 2

OpenStreetMap

Projekt OSM[1] [2] byl založen Stevem Coastem z Velké Británie v roce 2004. Cílem projektu je tvorba volně dostupných kartografických dat pro následné využití pro vizualizaci map, využití v navigacích apod. Na tvorbě dat se podílí komunita. Data lze zdarma využívat pod svobodnou licencí Open Data Commons Open Database License [3]. Licence pouze vyžaduje, aby bylo uvedeno autorství označením „© Přispěvatelé OpenStreetMap“.

2.1 Struktura dat

Veškerou geometrii v OSM tvoří dvě základní entity: uzly (node) a cesty (way) [7]. Každý uzel i cesta má své jednoznačné ID, obsahuje informaci o času vytvoření, uživatelské jméno autora a další. Bod obsahuje souřadnice vyjádřené v systému WGS-84. Uzly se mohou nacházet v mapě samostatně (např. název obce, pomník, památný strom . . .), nebo mohou být součástí cesty. Cesty spojují více uzlů do jednoho celku a tvoří buď polygon (pokud poslední uzel odpovídá prvnímu), např. budovy, lesy, jezera apod., nebo lomenou čáru (křivku), např. silnice, potok, hranice, apod.

Další entitou v OSM jsou relace (relation) [7], které sdružují více cest a uzlů do jedné entity.

Uzly, cesty i relace dále mohou obsahovat tagy [7], které se skládají z klíče a hodnoty, a slouží jako atributy pro určení typu a vlastností entity.

2.2 Hranice v České republice

Pro administrativní hranice jsou v OSM použity cesty, které jsou označeny tagem s klíčem `boundary` a hodnotou `administrative` [9]. Informaci o typu území (stát, kraj, obec, . . .), kterému jsou hranice určeny, definuje další tag s klíčem `admin_level`, jehož hodnota může nabývat čísel 1 až 10 (u některých zemí i 11). Číslo 1 je určeno pro území s nadnárodní vládou, ale v praxi není nikde taková hranice použita. Číslem 2 se označují národní hranice. Další hodnoty jsou závislé na konkrétní zemi, protože administrativní členění se v různých zemích liší. To je důvod, proč se v OSM používají pro rozlišení čísla, protože potom aplikace pro vykreslení těchto hranic nemusí znát, jaké administrativní členění je použito v různých zemích.

[1] OpenStreetMap

Na Wiki OSM [9] je možné dohledat, které hodnoty `admin_level` jsou v různých zemích použity a které hranice vyjadřují. Hodnoty, které se používají pro Českou republiku, znázorňuje tab. 2.1.

`admin_level`	**území**
2	stát
4	region soudržnosti
6	VÚSC[2]
7	okres
8	obec
10	katastrální území

Tabulka 2.1: Hodnoty `admin_level` pro ČR

Zdrojem dat pro administrativní hranice v OSM jsou katastrální mapy ČÚZK[3].

2.3 Data v XML

OSM mj. poskytují data ve formátu XML [22]. Soubory s daty OSM mají příponu `osm`. OSM poskytuje soubor `planet.osm` [16] s veškerými daty. K dispozici jsou také ze serverů partnerů soubory, které obsahují pouze data z určitého regionu (např. státu) [8] – ty ale bývají zpravidla aktualizovány s delší periodou, než globální soubor `planet.osm`.

2.4 Možný import hranic

Hranice z OSM by bylo možné importovat z regionálního XML souboru pro Českou republiku. Vyfiltrovaly by se relace, které obsahují tag s klíčem `boundary` a hodnotou administrative a tag s klíčem `admin_level` a hodnotou s požadovanou úrovní. Z těchto relací bychom získali jednotlivé cesty, které představují segmenty hranic a uzel, který v mapách slouží pro zobrazení názvu – z tohoto uzlu by bylo možné získat název územního celku. Nevýhodou tohoto postupu je, že XML soubor by program pro import dat musel procházet dvakrát, protože relace se nachází až na konci souboru. Z relace by se získaly ID cest a ID uzlu s názvem a ve druhém průchodu souborem by se nalezly tyto cesty a uzly. Druhou nevýhodou je fakt, že OSM neobsahuje pouze definici hranic, ale mnoho dalších geometrických dat a i soubor pouze s Českou republikou má velikost cca 15 GB.

[2] Vyšší územní samosprávný celek (dnešní kraje)
[3] Český úřad zeměměřický a katastrální

Kapitola 3

Registr územní identifikace, adres a nemovitostí

RÚIAN[1] je registr veřejné správy České republiky spravován ČÚZK[2] [13]. Jedná se o jeden ze čtyř základních registrů podle zákona č. 111/2009 Sb. Registr obsahuje referenční údaje o územních celcích a jednotkách včetně jejich vzájemných vazeb. Každý prvek obsahuje název, kód, definiční bod (souřadnice), hranice a vazby na další prvky (např. příslušnost katastrálního území k obci). Umožňuje zdarma dálkový přístup (VDP[3]) bez nutnosti registrace [11].

3.1 Členění

Registr obsahuje územní prvky a jednotky, které mají mezi sebou hierarchické vazby. Na nejvyšší úrovni je stát (Česká republika). U územních celků stát, regiony soudržnosti, VÚSC, okresy, obce, katastrální území a ZSJ[4] platí, že celkové území všech podřazených tvoří území nadřazeného celku. Dále registr obsahuje původní kraje[5], obce s rozšířenou působností, městské obvody apod. Na nejnižší úrovni obsahuje ulice, parcely, stavební objekty a adresní místa.

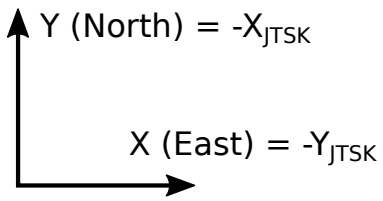
3.2 Výměnný formát

VDP umožňuje stáhnout data ve strojově čitelném formátu XML. K dispozici je mnoho souborů, přičemž každý obsahuje určitou část dat z registru, jako např. kompletní data, nebo pouze přírůstky, data pouze pro určitou část území (např. pro obec), s hranicemi nebo bez apod. Těchto souborů je velké množství, proto je k dispozici vyhledávací formulář pro jejich filtrování [10]. Název souboru odpovídá danému filtru, viz uživatelská příručka VDP [12].

[1] Registr územní identifikace, adres a nemovitostí
[2] Český úřad zeměměřický a katastrální
[3] Veřejný dálkový přístup k datům RÚIAN
[4] Základní sídelní jednotka
[5] Od 11. dubna 1960, zanikly vznikem VÚSC

3.3 Souřadnice

Souřadnice jsou v souborech výměnného formátu uloženy ve formátu S-JTSK, a to ve tvaru: `−Y −X`. S-JTSK je pravoúhlá souřadnicová síť, která se používá pro geodézii v České republice a na Slovensku. Hodnoty jsou záporné, protože je použit systém EPSG 5514. Číslování souřadnic znázorňuje obr. 3.1.



Obrázek 3.1: Znázornění číslování souřadnic v RÚIAN

3.4 Členění XML souboru

Soubor XML je rozdělen na dvě části – na hlavičku a data. Ve hlavičce jsou obsaženy informace o daném souboru – verze formátu, datum vytvoření, doba platnosti dat apod. V datové části se nachází jednotlivé územní celky ve skupinách podle svého typu seřazené od největších (státy, kraje, ...) po nejmenší (parcely, adresy, ...). Každý takový celek obsahuje svůj kód, název, kód nadřazeného celku, jehož je součástí, případně geometrii (definiční bod, hranice, ...) a další data podle typu územního celku a filtru dat.

3.5 Možný import hranic

RÚIAN obsahuje celé polygony hranic jednotlivých hranic. Navíc obsahuje oproti OSM mapám i základní sídelní jednotky – OSM mapy poskytují nejmenší územní celky katastrální území. Menší územní celky jsou výhodou, protože dovolují uživatelům přesnější definici vlastního území. Dále soubor s daty neobsahuje příliš mnoho dalších informací, tudíž je importovaný soubor oproti OSM podstatně menší – jeho velikost je cca 330 MB.

Kapitola 4

Databáze územních celků

Rozhodl jsem se využít data z RÚIAN a pro aplikaci použít územní celky typu kraj (VÚSC), okres, obec, katastrální území a základní sídelní jednotka (ZSJ). Výhodou tohoto výběru je fakt, že sloučením oblastí podřazených území dostaneme oblast nadřazeného území (např. pokud sloučíme všechny okresy v Jihomoravském kraji, získáme území tohoto kraje). Důvodem, proč nepoužít pouze jeden typ územního celku (např. ZSJ), je situace, kdy by uživatel v aplikaci mapu příliš oddálil – např. aby byla viditelná celá ČR. Potom by bylo třeba stáhnout ze serveru ohromné množství dat pro zobrazení všech území, s čímž by byl problém jak na straně serveru, který by musel jednomu uživateli naservírovat spoustu dat, tak na straně uživatele, který by potřeboval operační paměť s velkou kapacitou pro uložení těchto dat a výkonný procesor pro vykreslení všech viditelných hranic. Další možností, jak by bylo možné tento stav obejít, by bylo od určitého oddálení přestat území zobrazovat. To by ale mohlo vést ke zmatení uživatele, který by si mohl myslet, že aplikace nefunguje. Více typů územních celků navíc přidává tu výhodu, že pokud by uživatel chtěl definovat své území velké, může k jeho sestavení použít i větší územní celky, což mu urychlí práci oproti situaci, kdy by musel vybírat všechny jednotlivé ZSJ.

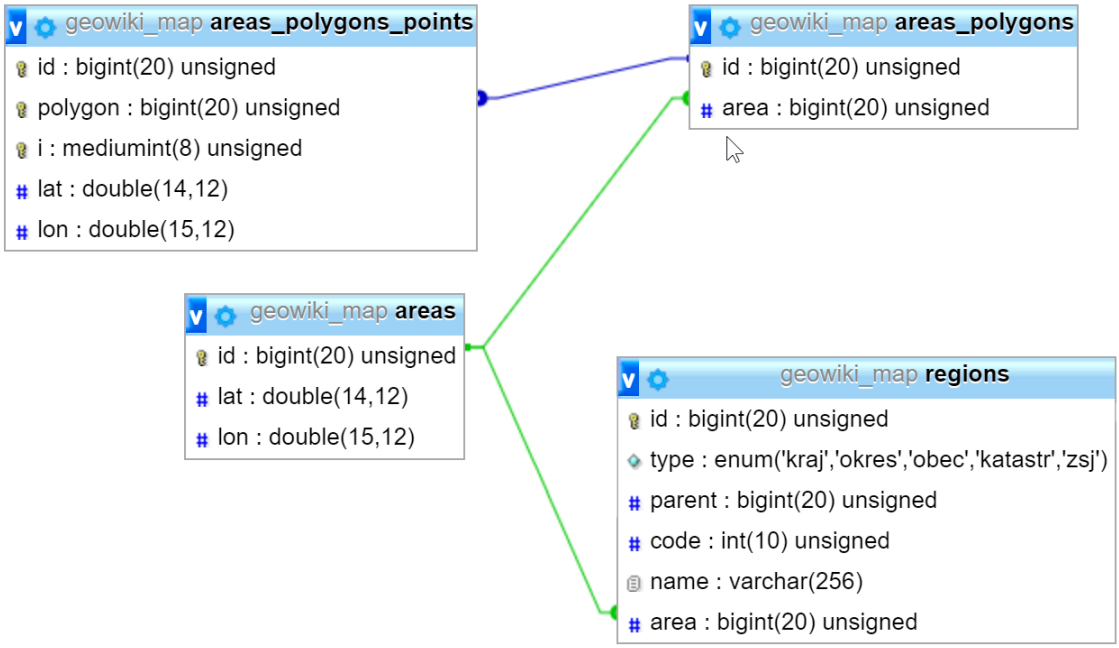
4.1 Tabulky pro územní celky

Návrh tabulek pro import územních celků znázorňuje obr. 4.1. Další tabulky a případně atributy do těchto tabulek potřebné pro vlastní aplikaci budou přidány v kapitole 6.

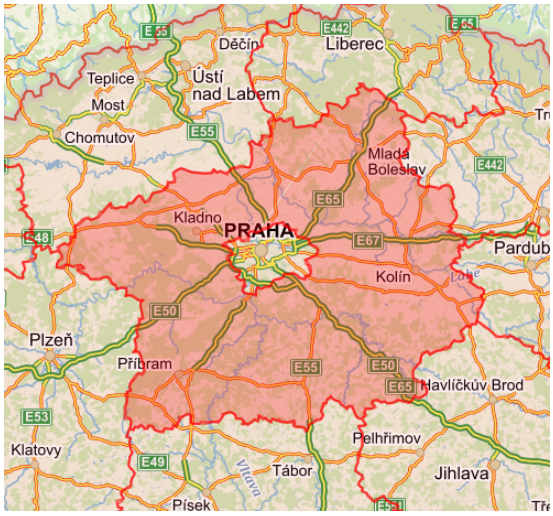
Územní celky (dále regiony) budou uloženy v tabulce `regions`. Atribut `type` určuje typ územního celku a atribut `parent` obsahuje ID nadřazeného regionu (např. region typu obec obsahuje ID příslušného regionu typu okres). Atribut `code` obsahuje kód územního celku použitého v registru RÚIAN a `name` název regionu (např. „Jihomoravský kraj“, „Brno“, „Královo pole“, ...).

Pro uložení hranic (oblastí) většiny regionů stačí obyčejný polygon. Ovšem v České republice existuje několik výjimek. Např. Středočeský kraj má ve svém území „díru“, protože se uvnitř něj nachází kraj Hlavní město Praha (viz obr. 4.2). Další případ je např. obec Tišnov, jejíž území se skládá ze tří oddělených oblastí (viz obr. 4.3). Uložení hranic pomocí jednoho polygonu je tedy nedostačující. Oblasti regionů tedy musí být definovány více polygony, které buď tvoří vnější, nebo vnitřní hranici („díru“).

Hlavní tabulkou pro oblast je tabulka `areas`, která obsahuje svoje ID a souřadnice `lat` (zeměpisná šířka) a `lon` (zeměpisná délka) v systému WGS-84, které později budou v aplikaci sloužit pro určení, ve které dlaždici se bude tato oblast nacházet. V tabulce



Obrázek 4.1: Tabulky pro územní celky



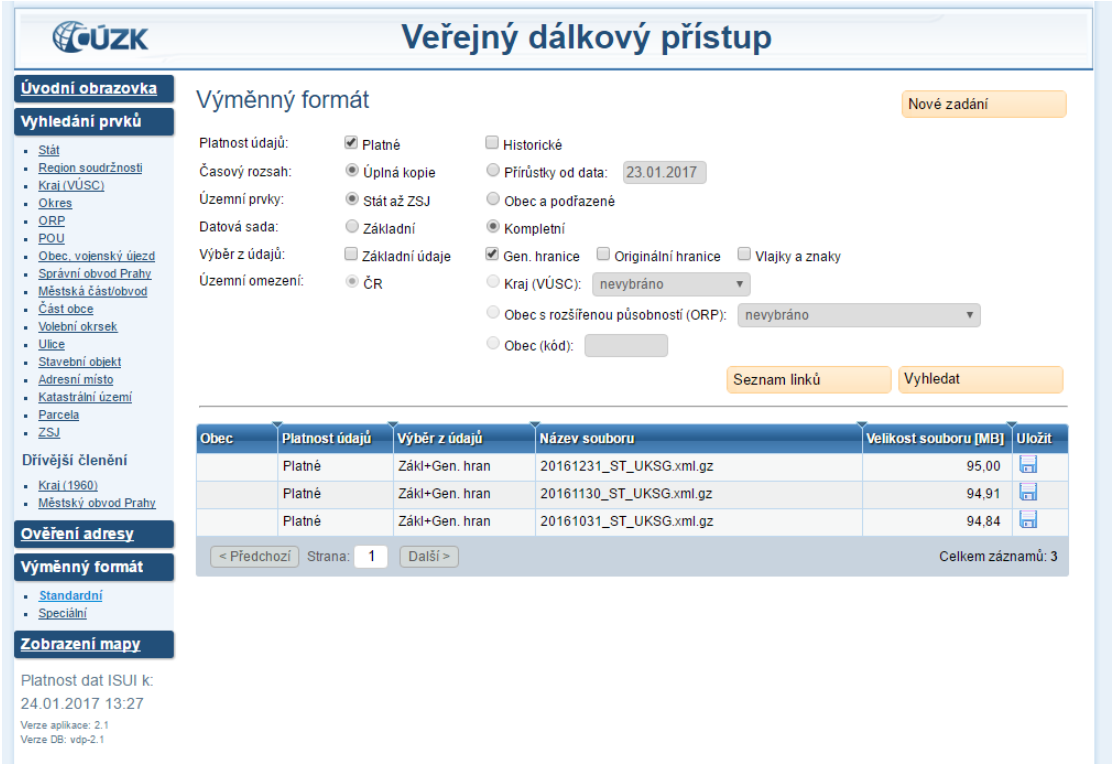
Obrázek 4.2: Oblast Středočeského kraje

Obrázek 4.3: Oblast obce Tišnov

`areas_polygons` jsou uloženy jednotlivé polygony. Obsahuje svoje ID, ID oblasti (atribut `area`). Způsob rozlišení, které území patří do oblasti definované více polygony, je popsán v kapitole 8.6. A na konec je tu tabulka `areas_polygons_points`, která obsahuje jednotlivé body polygonu. Každý bod má opět své ID a ID polygonu (atribut `polygon`). Atribut `i` určuje pořadí bodu a atributy `lat` a `lon` souřadnice bodu. Tabulka `regions` a `areas` jsou ve vztahu 1:1. Důvodem, proč jsem je nespojil do jedné tabulky, je, že později mohou být v databázi uloženy oblasti, které budou patřit jiným entitám, než pouze regionům.

4.2 Program pro import dat z RÚIAN

Program popsaný v této kapitole slouží pro import regionů a jejich hranic z registru RÚIAN do databázové struktury popsané v kapitole 4.1. Program pracuje s XML souborem získaným z VDP a s prázdnou databází, ve které vytvoří požadované tabulky a naplní je získanými daty. Jedná se o program pro jednorázové využití za účelem vytvoření základních regionů, které budou využívány aplikací. Program je napsaný v jazyce PHP [24] a pracuje s prázdnou MySQL [20] databází. Název souboru je součástí textu programu.



Obrázek 4.4: Vyplněný formulář pro nalezení soubor pro import dat z RÚIAN

4.2.1 Importovaný soubor

Vstupní XML soubor, se kterým tento program pracuje, je nutné získat z VDP. Program očekává soubor s platnými údaji, úplnou kopií, územními prvky „Stát až ZSJ“, kompletní datovou sadou a s generickými hranicemi. Screenshot na obr. 4.4 zachycuje vyplněný formulář pro vyhledání souboru s daty na webové stránce VDP.

Nalezené soubory by měly mít název ve tvaru `RRRRMMDD_ST_UKSG.xml.gz`, kde `RRRRMMDD` je datum vytvoření souboru. Pro získání souboru XML se musí stažený soubor rozbalit např. programem `unzip`, přejmenovat na `data.xml` a vložit ho do stejné složky, ve které se bude program spouštět.

4.2.2 Převod souřadnic

Registr RÚIAN používá souřadnicový systém S-JTSK (viz kapitola 3.3, ale návrh databáze (viz kapitola 4.1) počítá se souřadnicemi v systému WGS-84. Proto musí tento program provádět konverzi mezi těmito souřadnicovými systémy. Ke konverzi jsem použil JTSK coordinates converter [30] dostupný na GitHubu, ve kterém je k tomuto účelu využita metoda `JTSKtoWGS84` třídy `JTSK\Converter`.

4.2.3 Zpracování oblasti

Pro uložení hranic ve výměnném formátu registru RÚIAN je použit jazyk GML[1] [15], konkrétně struktura `MultiSurface`, která obsahuje struktury `Polygon` (jeden, či více – pokud se oblast regionu skládá z více částí). Definice hranice může vypadat např. takto:

```
<gml:MultiSurface gml:id="HZJ.116008"
                  srsName="urn:ogc:def:crs:EPSG::5514"
                  srsDimension="2"
                  xmlns:gml="http://www.opengis.net/gml/3.2">
  <gml:surfaceMember>
    <gml:Polygon gml:id="HZJ.116008.1">
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList>
            -696090.27 -1006723.53 -696087.47 -1006721.96
            -696062.01 -1006669.24 -696060.51 -1006666.14
            -696059.37 -1006663.77 (...)
          </gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:surfaceMember>
</gml:MultiSurface>
```

Ke zpracování této struktury slouží v programu funkce `parseBoundary`, která v parametru očekává objekt třídy `DOMNode` [27] elementu hranice územního celku – rodičovský element struktury `MultiSurface`. Tato funkce vytvoří v databázi oblast (tabulka `areas`) a následně projde strukturu `MultiSurface` a nalezne všechny vnější (`exterior`) a vnitřní (`interior`) části polygonů, které vloží do databáze (tabulka `areas_polygons`). Jednotlivé body se zpracovávají ve funkci `parseBoundary_coords`, která v parametru očekává textový obsah elementu `posList` (viz ukázka kódu definice hranice). Tato funkce převede jednotlivé souřadnice (dvojice hodnot) do systému WGS-84 a vrátí je v poli. Ve funkci `parseBoundary` se toto pole projde a jednotlivé body se vloží do databáze (tabulka `areas_polygons_points`). Na konci této funkce se určí souřadnice oblasti, které se

[1] Geography Markup Language

vypočítají jako střed opsaného obdélníku určeného z minimálních a maximálních hodnot zeměpisné šířky a délky.

4.2.4 Vlastní program

Na začátku programu se nejdříve vytvoří databázová struktura popsaná v kapitole 4.1. Následně se otevře soubor výměnného formátu pomocí třídy `XMLReader` [28] a v souboru se vybere tag `Data`, který obsahuje data o územních celcích. Územní celky jsou uspořádány do skupin podle svého typu a tyto skupiny jsou seřazeny podle hierarchie od nejnadřazenějších územních celků (první skupina obsahuje stát, poslední obsahuje základní sídelní jednotky). Výhodou tohoto uspořádání je, že k uložení všech závislostí (odkazů z podřazeného regionu na nadřazený) je možné provést v rámci jednoho čtení souboru, protože vždy, když se čtou data o podřazeném regionu, nadřazený už v tuto dobu byl přečten.

Uložení různých typů regionů do databáze se liší pouze hodnotou typu a na jaký typ nadřazeného regionu je odkaz. Ve výměnném formátu se různé typy liší jmény tagů a jmenných prostorů. Jinak jsou data pro účely tohoto programu stejně strukturovaná. Proto je pro zpracování všech typů regionů použit stejný úsek programu. Rozdíly mezi jednotlivými typy jsou uvedeny v proměnné `$areas`, ve které se nachází asociativní pole, kde klíčem je hodnota typu regionu v databázi a hodnotou je další asociativní pole, kde jsou uloženy konkrétní názvy tagů a jmenných prostorů pro konkrétní typ územního celku.

Pole v proměnné `$areas` je postupně procházeno a spouští se úsek programu pro zpracování skupiny územních celků. Asociativní pole v PHP zachovává pořadí svých prvků, tudíž jsou skupiny územních celků zpracovávány ve správném pořadí, jak jsou definovány v proměnné `$areas`, což odpovídá pořadí ve výměnném formátu.

V úseku programu pro zpracování skupiny územních celků se nachází cyklus, který postupně čte elementy územních celků, dokud se ve skupině nějaký nachází. U každého elementu projde subelementy a vytáhne z nich informace o daném regionu, jako je název, kód územního celku v RÚIAN a hranice, které se zpracují funkcí `parseBoundary`. Po získání veškerých informací vloží záznam do tabulky `regions`.

Odkazy na nadřazené regiony jsou vyhodnocovány již v tomto úseku kódu. Návrh databáze počítá s vlastními ID, které jsou generovány automaticky – tedy neodpovídají kódům z RÚIAN. Program si proto musí průběžně ukládat překladovou tabulku mezi kódy z RÚIAN a ID vygenerovaných v databázi. K tomuto účelu slouží proměnná `$idByCode`, ve které je pole, kde index odpovídá ID v databázi a hodnota kódu z RÚIAN. Toto pole je postupně plněno vždy po vložení nového regionu do databáze. Před vložením nového regionu do databáze (s výjimkou krajů, které nemají nadřazený region) je proveden překlad kódu z RÚIAN na ID nadřazeného regionu pomocí této tabulky.

4.2.5 Spuštění programu

Jelikož je program napsán v jazyce PHP, je nutné ho spouštět pomocí interpretu PHP. Soubor s programem je uložen pod názvem `transfer-data.php` ve složce `private`. Příkaz pro spuštění programu by mohl vypadat následovně:

```
php transfer-data.php
```

Data z registru RÚIAN jsou velmi velká, především co se týče údajů o hranicích jednotlivých územních celků. Proto může tento program běžet i několik hodin, než se všechna data zpracují a vloží do databáze.

Kapitola 5

Návrh aplikace

Aplikace by měla registrovanému uživateli s patřičnými opravněními poskytnout možnost definovat si vlastní území, a to sloučením území základních regionů jako jsou např. obce nebo katastrální území. Tato území se budou zobrazovat na interaktivní mapě, ve které je budou moci zobrazit i ostatní uživatelé. Mimo území definovaných ze základních regionů bude možné vytvořit na mapě i body (např. místa), trasy (např. ulice, stezky, ...) a plochy (budovy, pozemky, ...).

5.1 Základní regiony

Základní regiony jsou pro aplikaci převedeny ze státního registru RÚIAN, jedná se o regiony typu kraj (VÚSC), okres, obec, katastrální území a základní sídelní jednotka. Platí, že sloučením oblastí příslušných podřazených regionů vznikne oblast nadřazeného. Regiony bude možné v rámci aplikace rozdělovat (z jednoho regionu vzniknou rozdělením oblasti dva nové) a slučovat (spojením oblastí více regionů vznikne jeden nový). Rozdělovat bude možné pouze základní sídelní jednotky, protože rozdělením nadřazenějšího regionu by mohlo dojít k situaci, kdy by některý podřazený region měl částečně patřit prvnímu a částečně druhému nově vzniklému regionu. Slučovat bude možné pouze regiony stejného typu a patřící stejnému nadřazenému regionu – kdyby se spojili regiony patřící různým nadřazeným regionům, vznikla by opět situace, kdy by region měl z části patřit jednomu a z části jinému regionu. Také bude možné změnit, kterému nadřazenému regionu region náleží. U každé operace (rozdělení, spojení, změna nadřazeného) bude možné nastavit časový interval platnosti změny.

5.2 Uživatelská území

Uživatelská území bude možné definovat sloučením oblastí základních regionů. Není problém sloučit území různých typů, protože cílem sloučení je pouze získat území bez jakékoliv hierarchické návaznosti na základní typy regionů – území vyšších typů v tomto případě pouze slouží jako zjednodušení, aby uživatel nemusel vybírat všechny základní sídelní jednotky, ale mohl vybrat např. území celé obce. Body, trasy i plochy bude možné vytvořit kdekoliv na mapě nezávisle na základních regionech. Uživatelská území, body, trasy i plochy mohou mít definovanou dobu platnosti.

Dle pokynů vedoucího má být aplikace propojena s aplikací vytvořenou v rámci bakalářské práce *Zobrazování genealogických dat* [29], která obsahuje databázi farností, matrik

a obcí. Pro farnosti a matriky bude možné definovat uživatelská území, čímž se vytvoří propojení mezi databázemi jednotlivých aplikací a aplikace z uživatelského hlediska budou provázany hypertextovými odkazy. Obce budou provázané s příslušnými základními regiony a toto provázání bude možné upravovat z důvodu oprav a doplňování.

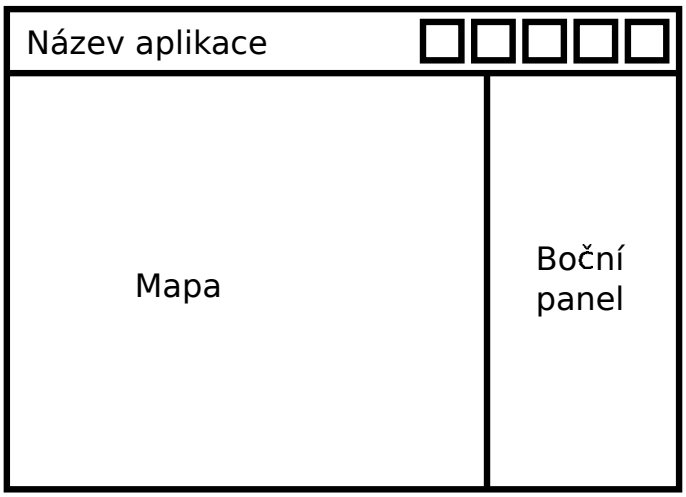
5.3 Uživatelé

Neregistrovaní uživatelé a uživatelé bez administrátorského oprávnění si budou moci pouze prohlížet vytvořená území. Registrovaní uživatelé s administrátorskými právy je budou moci upravovat, vytvářet a mazat.

Dle pokynů vedoucího mají mít obě aplikace jednotnou správu uživatelů. Moje aplikace bude navázána na uživatele v aplikaci na zobrazování genealogických dat, kde bude řešena registrace a přihlášení. Tato aplikace bude pouze detekovat, který uživatel je aktuálně přihlášený a jaká má oprávnění.

5.4 Uživatelské rozhraní

Hlavním prvkem v aplikaci bude mapa, ve které se budou jednotlivá území zobrazovat. V horní části obrazovky se bude nacházet lišta, ve které vlevo bude název aplikace a vpravo tlačítka s akcemi. Vpravo od mapy se bude nacházet boční panel, ve kterém se budou zobrazovat podrobnosti o aktuálně vybraném základním regionu nebo uživatelském území spolu s tlačítky pro operace s nimi – veškeré informace a formuláře pro úpravu se budou nacházet v tomto panelu. Základní rozložení aplikace znázorňuje obr. 5.1.



Obrázek 5.1: Návrh uživatelského rozhraní aplikace

Kapitola 6

Rozšíření databáze pro aplikaci

V kapitole 4 je popsána struktura databáze nezbytná pro převod dat o základních regionech z RÚIAN. V této kapitole bude popsáno rozšíření struktury databáze pro účely aplikace, která byla popsána v kapitole 5. V databázi je potřeba zohlednit časové platnosti jednotlivých základních regionů, uživatelská území, body, trasy a plochy včetně vzájemných vztahů mezi nimi.

6.1 Popis rozšíření stávajících tabulek a nových tabulek

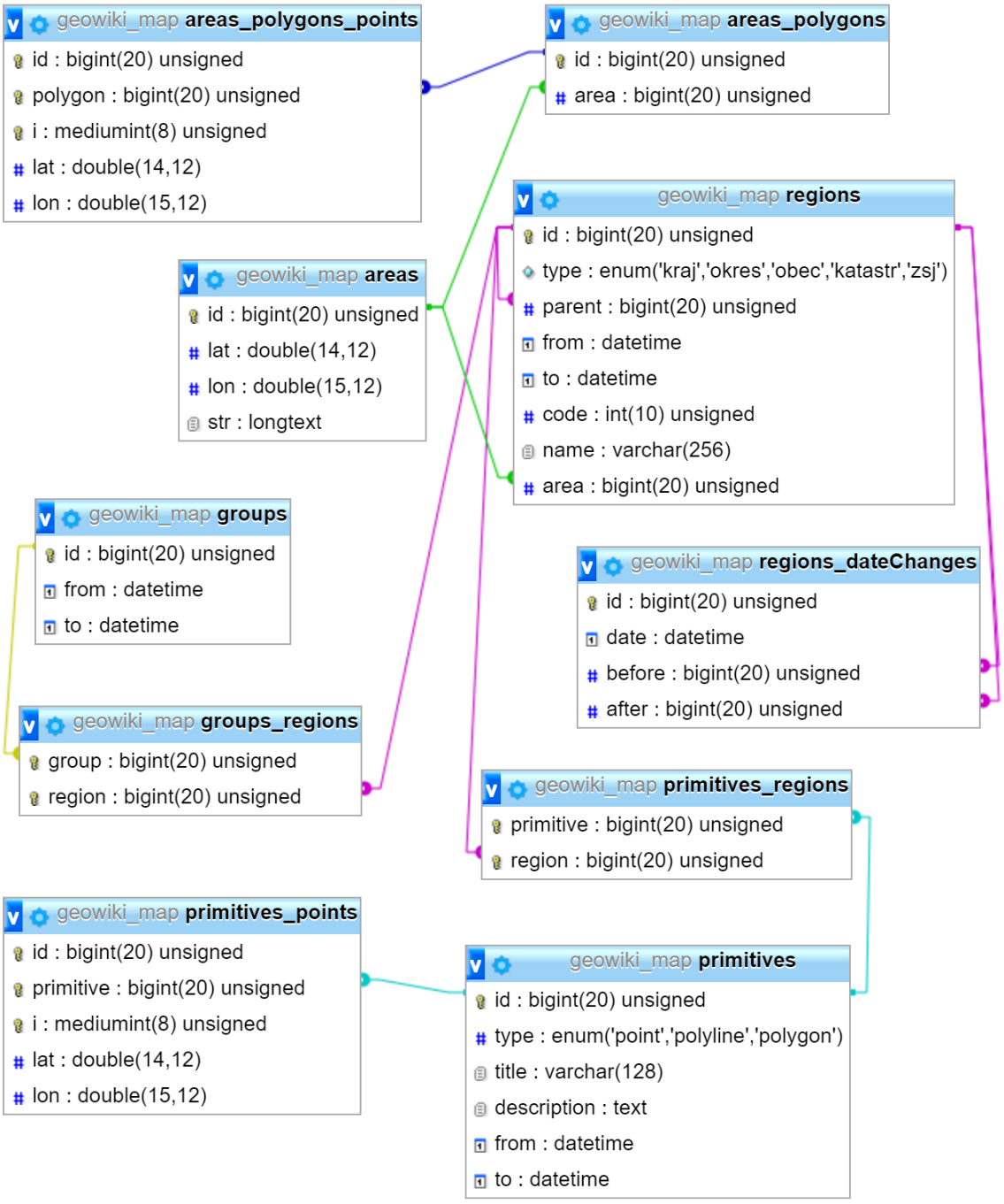
Návrh výsledné podoby databáze znázorňuje obrázek 6.1.

Tabulka `areas` byla rozšířená o atribut `str`, do kterého se bude ukládat geometrická definice oblasti serializována do řetězce. Důvodem je optimalizace při načítání hranic základních regionů, protože je výpočetně méně náročné získat hranice z takto serializovaných dat, než procházet tabulky `areas_polygons` (řádově tisíce záznamů) a `areas_polygons_points` (řádově miliony záznamů).

Tabulka základních regionů `regions` byla rozšířena o atributy `from` a `to`, které udávají časovou platnost regionů a jejich hranic. Obě hodnoty mohou obsahovat hodnotnu `NULL`, která znamená, že doba platnosti není definována – představuje časový ekvivalent ∞, respektive $-\infty$.

Pokud časová hranice platnosti nemá hodnotu `NULL`, musí existovat minimálně jeden region, který danému regionu předcházel, respektive následuje po něm, protože by neměla vzniknout situace, že by základní region vznikl z ničeho, nebo by zanikl, aniž by jeho území vyplnil jiný region. Jelikož region mohl vzniknout např. spojením dvou a tudíž předcházející regiony daného regionu jsou dva (obecně libovolné množství), není možné tuto informaci uložit do tabulky `regions`, ale musí být uložena ve vlastní tabulce. K tomuto účelu slouží tabulka `regions_dateChanges`, která obsahuje kromě identifikátoru atributy `date` (čas, ve kterém došlo ke změně), `before` (ID regionu, který předcházel změně) a `after` (ID regionu, který vznikl po této změně). Když se vrátíme k příkladu, kdy sloučením dvou regionů vznikl jeden nový, tato změna bude tabulce `regions_dateChanges` reprezentována dvěma záznamy, které se budou lišit pouze v atributu `before`.

Pro uložení uživatelských území slouží tabulka `groups`, která obsahuje ID a atributy `from` a `to`, které jsou nepovinné – mohou obsahovat hodnotu `NULL` podobně jako u základních regionů. Ovšem u uživatelských území neplatí, že jejich vzniku muselo něco předcházet, nebo následovat po jejich zániku, proto v databázi není žádná tabulka, která by reflektovala změny podobně jako je to u základních regionů v tabulce `regions_dateChanges`. Pro při-



Obrázek 6.1: Tabulky pro aplikaci

řazení základních regionů k uživatelským územím slouží tabulka `groups_regions`. Jedná se čistě o propojovací tabulku, proto neobsahuje žádné ID, pouze atributy `group` a `region` obsahující ID uživatelských území a základních regionů.

Pro body, trasy i plochy slouží jediná tabulka `primitives`, a to z toho důvodu, že všechy tyto entity mají podobné vlastnosti. Tabulka `primitives` obsahuje ID, atribut `type`, který rozlišuje, o kterou z těchto tří entit se jedná, atributy `title`, `description` (název a popis), které jsou volitelné, a atributy `from` a `to` (časová platnost), které mají stejný význam, jako u uživatelských území. Geometrie těchto entit je uložena v tabulce `primitives_points`, která obsahuje jednotlivé body, ze kterých se daná entita skládá. Tabulka opět obsahuje ID, potom atribut `primitive` s ID entity, které daný body patří, atribut `i`, který udává pořadí bodu, a atributy `lat` a `lon`, které udávají souřadnice bodu v souřadném systému WGS-84. Trasy i plochy jsou definovány uspořádanou množinou bodů, tudíž jsou pro tyto entity tabulky `primitives` a `primitives_points` ve vztahu 1:N. Bod je definován jediným bodem, a proto pro něj platí, že tyto tabulky jsou ve vztahu 1:1. Na konec je tu tabulka `primitives_regions`, což je propojovací tabulka mezi těmito třemi entitami a základními regiony – obsahuje pouze atributy `primitive` a `region` s identifikátory. Tato tabulka slouží pro určení, do kterého území základních regionů geometrie dané entity spadá. Tato informace bude v aplikaci využita pro zobrazení seznamu všech bodů, tras a ploch, které se budou nacházet v aktuálně vybraném základním regionu.

6.2 Program pro vytvoření řetězců pro oblasti

Soubor ve složce `private` s názvem `create-area-strings.php` obsahuje program, který do databáze vloží serializovanou podobu geometrických definic oblastí.

Program nejprve vytvoří patřičný atribut `str` v tabulce `area` – dočasně s povolenou hodnotou `NULL` (výchozí hodnota – pro oblasti, pro které ještě nebyl vytvořen řetězec). Následně projde všechny oblasti v databázi a pomocí třídy `MultiPolygon` (kapitola 8.6) serializuje jejich hranice do podoby řetězce, který uloží do databáze. Na konec program zakáže v atributu `str` hodnotu `NULL` – všechny oblasti již byly serializovány, tudíž žádný záznam v tomto atributu neobsahuje hodnotu `NULL`.

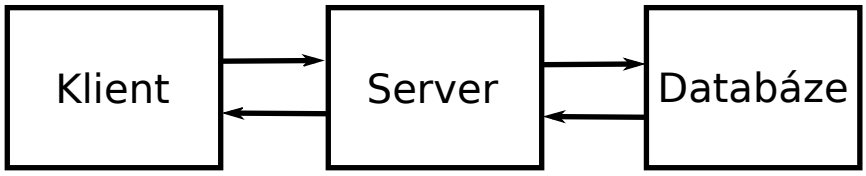
6.3 Program pro rozšíření databáze

Program obsažený v souboru `extend-db.php`, který se také nachází ve složce `private`, obsahuje pouze spuštění SQL příkazů, které rozšíří databázi tak, jak je popsáno v kapitole 6.1. Jediné rozšíření, který tento program neudělá, je vytvoření atributu `str` v tabulce `areas`, protože tuto činnost již provádí program `create-area-strings.php`.

Kapitola 7

Serverová část aplikace

I tato aplikace jako většina webových aplikací je navržena jako třívrstvá – třívrstvou architekturu znázorňuje obrázek 7.1. Aplikace se skládá ze tří vrstev: datová (databáze – popsána v kapitolách 4 a 6), serverová (o té pojednává tato kapitola) a klientská (běžící v prohlížeči uživatele - o té bude kapitola 9). Každá z těchto vrstev může běžet na jiném stroji. Serverová část slouží především k oddělení klientské vrstvy od datové. Logika, která se nachází v serverové části, by mohla být i součástí klientské vrstvy, přičemž by klientská vrstvá komunikovala přímo s datovou. To by ale představovalo bezpečnostní riziko, protože by toto řešení vyžadovalo veřejný přístup k databázi a potom by mohl kdokoliv napsat vlastní program, který by pracoval s databází po svém a mohl tím obejít veškerá omezení definována v originální aplikaci.



Obrázek 7.1: Třívrstvá architektura

Serverová část tedy zpracovává požadavky od klienta, kontroluje jejich validitu a pokud je požadavek oprávněný, projeví změny v databázi, nebo získá data z databáze a pošle je zpět klientovi. Nachází se zde veškerá logika, která je k těmto účelům nutná.

Pro vytvoření serverové části jsem zvolil stejně jako u programů pro vytvoření databáze jazyk PHP [24], který je v této oblasti velmi oblíbený. Dále se v serverové části nachází konfigurační soubory `.htaccess` pro webový server Apache [1], pomocí kterých je nastaveno routování a přístup k souborům přes webový protokol HTTP(S).

7.1 Třída pro práci s databází

Práce s databází je postavena na knihovně PDO [23]. Pro usnadnění je v aplikaci vytvořena třída `Database` ve jmenném prostoru `App\Utils`, která je obálkou (adaptérem) nad objektem třídy `PDO`.

Obsahuje metody `exec` a `query`, které provedou SQL dotaz. Pokud dotaz selže, vyvolá se výjimka. Při úspěchu metoda `exec` vrací počet ovlivněných řádků a metoda `query` objekt

třídy `PDOStatement`, se kterým je možné dále pracovat. Pro SQL dotaz `SELECT` jsou určeny metody `value`, `row`, `col` a `table`, které se liší formou, ve které vrací výsledek dotazu. Metoda `value` vrátí pouze hodnotu prvního atributu prvního záznamu. Pokud dotaz nevrátil žádné záznamy, tato metoda vrátí hodnotu `false`. Metoda `row` vrátí první záznam v asociativním poli. Pokud dotaz nevrátil žádné záznamy, také vrátí hodnotu `false`. Metoda `col` vrátí v poli první atributy ze všech záznamů. Pokud není žádný záznam, vrátí prázdné pole. A metoda `table` vrátí pole všech záznamů – každý záznam je v asociativním poli jako u metody `row`. Když dotaz nevrátí žádný záznam, také vrátí prázdné pole. Všechny čtyři metody pro svoji funkci využívají metodu `query`.

Vlastnost `lastId` obsahuje poslední identifikátor, který byl automaticky vytvořen po SQL příkazu `INSERT` auto-inkrementem. Tato vlastnost je navázána na metodu `lastInsertId` objektu třídy `PDO`.

Pro transakce obsahuje třída metody `begin` (zahájení transakce), `commit` (potvrzení transakce) a `rollback` (vrácení provedených změn). Přidanou hodnotou těchto metod je možnost vnořování více transakcí do sebe. Pokud ještě nebyla zahájena transakce, metoda `begin` ji zahájí, jinak provede SQL příkaz `SAVEPOINT`. Metoda `commit`, pokud je zavolána v první úrovni, provede SQL příkaz `COMMIT`, jinak uvolní poslední bod příkazem `RELEASE SAVEPOINT`. Metoda `rollback` vrátí změny provede od začátku transakce, respektive od příslušného bodu. Vnořování transakcí je výhodné při zásobníkovém volání funkcí, které obsahují SQL příkazy.

7.2 Vlastnosti objektů

V jazyce PHP se počítané vlastnosti definují pomocí metod `__get` a `__set`. To je nepraktické, pokud objekt má takových vlastností hodně, protože všechny musí být definovány v těchto dvou metodách. Proto do tříd vkládám trait `Properties`, který se nachází ve jmenném prostoru `App\Utils`, který definuje metody `__get` a `__set` tak, že zavolá příslušnou metodu, která začíná slovem `get`, respektive `set`. Tudíž pro definici vlastnosti s názvem `mojeVlastnost` stačí definovat funkce `getMojeVlastnost` a `setMojeVlastnost` (případně pouze jednu z nich, pokud má být vlastnost pouze pro čtení nebo pouze pro zápis). Pokud se bude definovat další vlastnost, budou to další dvě nové metody – tím se oddělí logika jednotlivých vlastností. Pro úplnost trait `Properties` definuje i metodu `__call`, která zajístí stejné metody i pro veřejné členské proměnné třídy.

7.3 Inicializace

Vložením souboru `init.php`, který se nachází ve složce `php`, se zajistí, že v globální proměnné `$db` se bude nacházet instance třídy `Database` s připojením do databáze. Připojení k databázi se nachází v souboru `db.php`.

Dále se zajistí, že všechny třídy pro serverovou část aplikace se vloží, jakmile jsou potřeba. To je zajištěno voláním funkce `spl_autoload_register` [26], které se nachází v souboru `autoload.php`.

Na konec se do proměnné `$role` uloží role aktuálně přihlášeného uživatele. Pokud není přihlášen žádný uživatel, proměnná bude obsahovat hodnotu `null`. Správa uživatelů je řešena v aplikaci vytvořené v rámci bakalářské práce *Zobrazování genealogických dat* [29]. V této aplikaci je použit Nette Framework [21], ve kterém je přihlašování uživatelů řešeno příslušnou knihovnou, která údaje o aktuálně přihlášeném uživateli ukládá do sessions [25].

V souboru `role.php`, který je vkládán v souboru `init.txt`, se nachází definice funkce, která vytahuje údaje ze sessions a vrací roli uživatele, která je tam uložena. Díky této informaci tato aplikace určuje, jaké akce je aktuální uživatel oprávněn provádět.

Vložením souboru `init.php` se zajistí všechny výše zmíněné zdroje. Tento soubor je určen, aby byl vložen na začátku každého PHP souboru, který je přístupný přes webové rozhraní, aby se v daném souboru řešila pouze logika, kterou má daný soubor provádět, a nemuselo se v rámci tohoto souboru řešit získávání zdrojů.

7.4 Třídy entit

Ve jmenném prostoru `App` se nachází třídy tří hlavních entit, které se nachází v databázi. Třída `Region` představuje základní region, třída `Group` představuje uživatelské území a třída `Primitive` představuje entity uložené v tabulce `primitives`: bod, trasu, nebo plochu.

Všechny tři třídy jsou vytvořeny tak, že mají soukromý konstruktor, ve kterém se načítají data z hlavní tabulky entity (`regions`, `groups` nebo `primitives`). Instance těchto tříd se získávají pomocí statických metod `get`, která ve svém parametru přijímá identifikátor entity. Tato metoda buď vrátí instanci třídy – pokud je předaný identifikátor platný, nebo hodnotu `false`, pokud se záznam v databázi nenachází. Pokud už byla instance s daným ID vytvořena, při opakovaném volání se vrátí stejná instance. K tomuto účelu se vytvořené instance uchovávají v soukromém statickém poli `$loaded`.

Data z ostatních tabulek patřící entitě jsou načítána metodou „lazy-loading“ – tzn. že jsou načtena až v době, kdy jsou potřeba – jakmile je k těmto údajům přistoupeno (např. přečtením hodnoty vlastnosti objektu, nebo zavoláním metody pracující s těmito daty).

Nové entity jsou vytvářeny pomocí statické metody `create`. Atributy těchto entit se předávají pomocí parametrů těchto metod – v každé třídě má tato metoda jiné parametry v závislosti na tom, o jakou entitu se jedná. Entity se z databáze odstraňují zavoláním metody `delete`, která zajistí, že se odstraní i všechny závislosti dané entity.

Úpravy atributů entity včetně dat v závislých tabulkách jsou řešeny pomocí vlastností. Data se v databázi neaktualizují po každé jednotlivé změně vlastnosti, protože se předpokládá, že při jedné akci od uživatele se může změnit více atributů zároveň. Při změně vlastnosti se tedy pouze aktualizuje daná hodnota v instanci a nastaví se příznak, že data v instanci byla změněna, ale ještě nebyla uložena do databáze. Pro uložení do databáze je potřeba zavolat metodu `save`.

7.5 Komunikace s klientskou částí

Výměna dat mezi klientskou a serverovou částí probíhá tak, že z klientské části aplikace se GET nebo POST metodou protokolu HTTP zavolá soubor PHP, který daný požadavek zpracuje a odpověď vygeneruje ve formátu JSON [6]. Tyto soubory jsou uloženy ve složce `json` a mají příponu `.json.php`, ale z klientské části jsou volány pouze s příponou `.json`. Přesměrování na správný název souboru je definováno v souboru `.htaccesss` určený pro webový server Apache.

Tyto soubory zpracovávají různé požadavky od klientské části aplikace – některé slouží pouze pro získání dat z databáze, některé zase pro uložení dat, nebo mazání. V těchto souborech jsou ošetřena odeslaná data a pokud jsou validní a aktuálně přihlášený uživatel má patřičná oprávnění, je provedena požadovaná akce a na výstup je vytištěn výsledek ve formátu JSON.

Nacházejí se zde tyto soubory:

regions.json.php – Získá data o základních regionech a jejich hranice, které mají střed uprostřed čtverce, jehož souřadnice se očekávají v parametrech požadavku na tento soubor. Velikost čtverce je závislá na požadovaném typu regionu (typ je také očekáván v parametru) – tento soubor vždy vrací základní regiony o určitém typu. Tato data slouží pro zobrazování regionů na mapě.

region-groups-and-primitives.json.php – Získá seznam všech uživatelských území, bodů, tras a ploch nacházejících se uvnitř daného základního regionu – identifikátor regionu se očekává v parametru požadavku. Data slouží pro zobrazení těchto entit v detailu regionu.

parents-select.json.php – Získá seznam základních regionů, které je možné nastavit jako nadřazený region daného regionu – jeho identifikátor je očekáván v parametru požadavku. Toto je určené pouze pro uživatele s administrátorským oprávněním pro změnu nadřazeného regionu v editaci základního regionu.

region-date-limits.json.php – Získá minimální a maximální hodnotu časového omezení základního regionu, které lze nastavit. Identifikátor regionu je očekáván v parametru požadavku. Určeno pro uživatele s administrátorským oprávnění pro editaci základního regionu.

region-change-parent-areas.json.php – Získá hranice všech základních regionů, u kterých je nutné změnit oblast při změně nadřazeného regionu. Identifikátory měněného a nového nadřazeného regionu jsou očekávány v parametrech požadavku. Také určeno pro editaci základního regionu.

save-region.json.php – Uloží změny základního regionu do databáze – pokud na tuto akci má uživatel patřičná oprávnění.

save-join-regions.json.php – Uloží do databáze rozdělení základního regionu na dva nové. Určeno pouze pro uživatele s administrátorským oprávněním.

save-split-regions.json.php – Uloží do databáze spojení více základních regionů do jednoho nového. Určeno pouze pro uživatele s administrátorským oprávněním.

select-region.json.php – Uloží do databáze propojení základního regionu a obce v bakalářské práci *Zobrazování genealogických dat* [29]. Pouze pro uživatele s administrátorským oprávněním

group-data.json.php – Získá data o uživatelském území, jeho identifikátor je očekáván v parametru požadavku. Slouží pro zobrazení detailu o uživatelském území.

save-group.json.php – Slouží pro uložení nového, nebo editaci již existujícího uživatelského území. Pouze pro uživatele s administrátorským oprávněním.

delete-group.json.php – Slouží pro smazání uživatelského území z databáze. Identifikátor daného území je očekáván v parametru požadavku. Určeno pouze pro administrátory.

primitive-data.json.php – Získá data o bodu, trase, nebo ploše. Identifikátor dané entity je očekáván v parametru požadavku. Slouží pro zobrazení detailu o entitě.

save-primitive.json.php – Slouží pro uložení nového, nebo editaci již existujícího bodu, trasy, nebo plochy. Pouze pro uživatele s administrátorským oprávněním.

delete-primitive.json.php – Slouží pro smazání uživatelského území z databáze. Identifikátor daného území je očekáván v parametru požadavku. Určeno pouze pro administrátory.

7.6 Propojení aplikací

Propojení mezi touto aplikací a aplikací vytvářenou v rámci bakalářské práce *Zobrazování genealogických dat* [29] se v serverové části nachází ve třídách `Group` a `Region`.

Třída `Group` (uživatelská území) je propojena s databází aplikace na zobrazování genealogických dat s farnostmi a matrikami – propojení je vytvořeno za účelem zobrazení území dané farnosti nebo matriky na mapě. Po technické stránce je propojení řešeno tak, že v příslušných tabulkách farností a matrik se nachází cizí klíč směřující do databáze této aplikace do tabulky `groups`. Třída `Group` obsahuje navíc další vlastnosti, které obsahují identifikátory a názvy farnosti nebo matriky. Tyto údaje načítané metodou „lazy-loading“ stejně jako ostatní data z vedlejších tabulek. Název farnosti nebo matriky se využívá v klientské části aplikace.

Třída `Region` je propojena s obcemi v databázi aplikace na zobrazování genealogických dat. Opět se v příslušné tabulce obcí v této databázi nachází cizí klíč směřující do databáze této aplikace – v tomto případě do tabulky `regions`. Třída `Region` obsahuje navíc vlastnost `obec_ID`. Vlastnost pro název se zde nenachází, protože se předpokládá, že se bude shodovat.

Tyto údaje, které přidává propojení aplikací, jsou promítnuty i v komunikaci mezi serverovou a klientskou částí, která byla popsána v podkapitole 7.5.

Kapitola 8

Práce s polygony

Jednotlivé hranice území jsou v databázi uloženy jako polygony a stejným způsobem se také zobrazují v klientské části aplikace. Je tedy vhodné tyto polygony reprezentovat odpovídající datovou strukturou. Dále v aplikaci je možné hranice upravovat, proto je nutné implementovat pomocné funkce pro operace s těmito strukturami, jako např.: sjednocení, průnik, rozdíl apod.

Pro tyto účely je vytvořeno několik tříd, které jsou implementovány v jazycích PHP (pro serverovou část) a JavaScript (pro klientskou část).

8.1 Třída `Compare`

Jednotlivé složky souřadnic bodů jsou uloženy jako čísla s plovoucí desetinnou čárkou. Při operaci s těmito čísly může docházet k nepřesnostem a proto operace porovnání na rovnost nemusí vždy fungovat i přesto, že by z matematického hlediska měla. Tento fakt může u některých algoritmů pro operace nad polygony způsobovat problémy. Tento nedostatek se snaží řešit třída `Compare`, která obsahuje metody `equals` (rovnost), `greater` (větší než), `greaterOrEquals` (větší než nebo rovno) atd. Tyto metody počítají s určitou tolerancí, kterou je možné v instanci třídy nastavit, a to tak, že pokud rozdíl mezi čísly je menší, než hodnota této tolerance, považují se tyto dvě čísla za rovná.

8.2 Třída `XY`

Tato třída obsahuje souřadnice ve dvourozměrném kartézském souřadném systému. Je možné ji použít pro uložení absolutní pozice bodu, nebo jako vektor. Kromě složek `x` a `y` obsahuje také počítané vlasnosti `length` představující vzdálenost bodu od počátku (absolutní hodnota vektoru) a `dir` představující směr vektoru – úhel otočení od vektoru $(1, 0)$ proti směru hodinových ručiček. Obsahuje metody `add` (sečtení), `sub` (odečtení), `mul` (vynásobení) atd. pro operaci se dvěma objekty této třídy. Tyto metody provedou danou operaci nad oběma složkami `x` a `y`.

8.3 Třída `Angle`

Třída `Angle` může představovat úhel mezi dvěma vektory, nebo také směr vektoru – úhel mezi daným vektorem a vektorem $(1, 0)$. Instance vnitřně ukládá velikost úhlu ve stupních. To je výhodné z důvodu, že např. úhly jako pravý ($90\,^\circ$), přímý ($180\,^\circ$) nebo plný ($360\,^\circ$) se

dají ve stupních vyjádřit jako celá čísla. V radiánech jsou hodnoty těchto úhlů iracionální čísla, která v datovém typu číslo s plovoucí desetinnou čárkou nelze přesně vyjádřit. Tato hodnota se pokaždé operaci upraví tak, aby spadala do intervalu $\langle 0; 360)$.

Přesto, že vnitřně tato třída ukládá hodnotu ve stupních, je možné s ních pracovat se stupni i radiány, jelikož pro vnější rozhraní obsahuje jak vlastnost `degrees` (stupně), tak i vlastnost `radians` (radiány), při jejímž čtení nebo zápisu se provádí převod mezi stupněmi a radiány.

Obsahuje metody pro sčítání a odečítaní a metodu pro porovnání dvou úhlů.

8.4 Třída `Line`

Tato třída představuje orientovanou úsečku – obsahuje dva objekty třídy `XY` – počátek a konec. Tato třída není pro ukládání dat v aplikaci důležitá, protože v aplikaci se pracuje primárně s polygony, které se skládají z bodů – tato třída se používá v algoritmech pro operaci s polygony, kde je takováto struktura často užitečná. Navíc sama obsahuje několik metod pro práci s úsečkami, které jsou dále využívány.

Metoda `contains` zjistí, zda se daný bod předaný v prvním parametru nachází na úsečce. Metoda `isParallel` zkontroluje, zde jsou dvě úsečky vůči sobě rovnoběžné. A metoda `onLine`, zda leží na stéjné přímce.

Nejdůležitější metodou je metoda `intersection`, která vypočítá průsečík dvou úseček. Výsledkem může být hodnota `null`, pokud se úsečky neprotínají, bod, pokud se protínají a jsou různoběžné, nebo úsečka, pokud se protínají a jsou rovnoběžné.

8.5 Třída `Polygon`

Polygon (mnohoúhelník) je v této třídě definnován jako posloupnost bodů (objekty třídy `XY`). Polygon může být nekonvexní a nezáleží, zda jsou body definovány ve směru, nebo proti směru hodinových ručiček. Je možné, že se v polygonu mohou některé strany křížit. Některé algoritmy si neumí s takto definovaným polygonem poradit, proto tu je metoda `optimize`, která tento problém vyřeší. Odstraní nadbytečné body – pokud více po sobě jdoucích bodů leželo ve stejné přímce, nebo pokud úhel u některého vrcholu byl roven $360\,^\circ$. Touto redukcí může dojít k odstranění všech bodů (polygon byl definován tak, že měl nulový obsah) – v tom případě tato metoda vrátí hodnotu `null`. Pokud se některé strany polygonu křížili, tato metoda polygon v těchto místech rozdělí a vrátí objekt třídy `MultiPolygon`, viz 8.6. V ostatních případech vrátí novou instanci třídy `Polygon` – původní instance zůstane zachovalá.

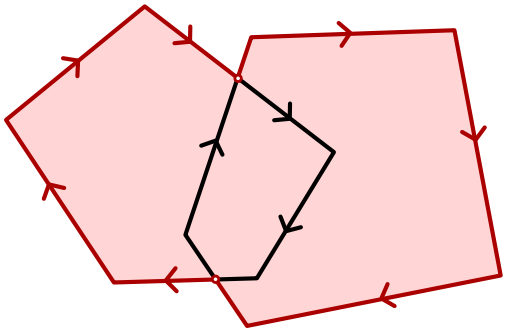
Třída obsahuje počítané vlastnosti `lines` (pole objektů třídy `Line` – jednotlivé strany polygonu), `leftTop` (objekt třídy `XY` obsahující minimální hodnoty složek `x` a `y` bodů polygonu), `rightBottom` (maximální hodnoty složek `x` a `y`), `size` (rozdíl hodnot `rightBottom` a `leftTop`) a `center` (aritmetický průměr hodnot `leftTop` a `rightBottom`). Dále obsahuje metody pro manipulaci `move` (posun polygonu o daný vektor) a `scale` (jednotlivé body polygonu se vynásobí jako vektory daným koeficientem).

Vlastnost `isClockwise` určuje, zda jsou definiční body polygonu uspořádány ve směru hodinových ručiček. Některé algoritmy očekávají tuto vlastnost a proto třída má metodu `reversePoints`, která vrátí novou instanci třídy `Polygon` s obráceným pořadím bodů.

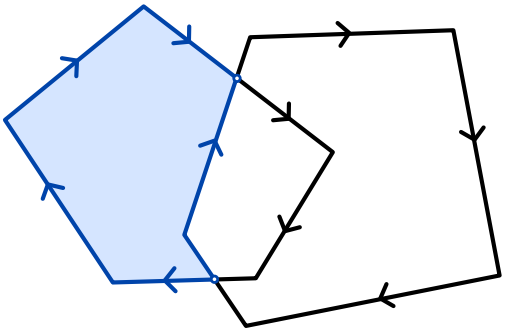
Metoda `contains` určuje, zda se daný bod nachází uvnitř polygonu.

A nakonec třída obsahuje metody `union` (sjednocení dvou polygonů), `sub` (rozdíl dvou polygonů), `penetration` (průnik dvou polygonů) a `diff` (vyloučení dvou polygonů). Algoritmy prvních tří zmíněných jsou velmi podobné, proto jsou ve třídě definovány jedinou soukromou metodou `operation`, která jako parametr dostává název operace. Tato metoda je potom volána ze zmíněných veřejných metod. Uvnitř metody `operation` je rozdílné chování odděleno pomocí podmínek. Samotný algoritmus této metody se skládá ze dvou částí. Nejdříve se naleznou všechny průsečíky mezi jednotlivými stranami obou polygonů, přičemž se vyfiltrují ty, které se pro sestavení výsledného polygonu nepoužijí. Tento úkol obstarává další soukromá metoda `intersectionPoints`.

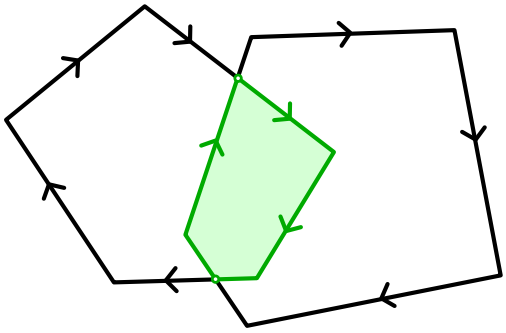
Druhá část algoritmu prochází jednotlivými body polygonů tak, že začne v určitém bodu prvního polygonu a inkrementuje se do té doby, dokud se k němu opět nevrátí. Pokud program narazí na průsečík, skočí do druhého polygonu a inkrementuje přes jeho body. Pokud tam nalezne další průsečík, vrací se opět do prvního. Jednotlivé operace (sjednocení, rozdíl a průnik) se liší místem počátku a směrem průchodu druhým polygonem. Operace sjednocení začíná v bodě prvního polygonu, který se nachází mimo druhý polygon a oba polygony prochází po směru hodinových ručiček (viz obrázek 8.1). Při operaci rozdílu se začíná stejným bodem, jako u sjednocení, ale druhý polygon se prochází opačným směrem (viz obrázek 8.2). Při operaci průniku se začíná v bodě prvního polygonu, který se nachází uvnitř druhého, jinak je algoritmus stejný jako u sjednocení (viz obrázek 8.3).



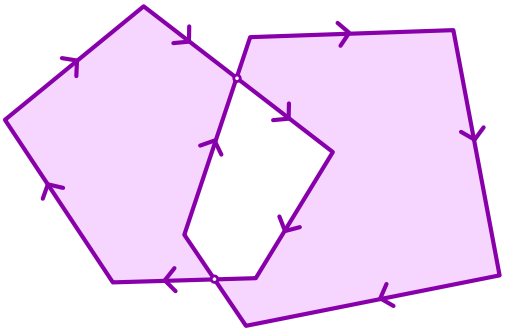
Obrázek 8.1: Sjednocení dvou polygonů



Obrázek 8.2: Rozdíl dvou polygonů



Obrázek 8.3: Průnik dvou polygonů



Obrázek 8.4: Vyloučení dvou polygonů

Pokud polygony nemají žádné průsečíky, mohlo k tomuto stavu dojít ve dvou případech. Buď se nijak neprotínají (nemají žádný průnik), nebo se jeden nachází uvnitř druhého. O který z těchto stavů se jedná, se zjistí voláními metody `contains`. Výsledek zavisí na

typu operace. Např. výsledek sjednocení v prvním případě bude objekt třídy `MultiPolygon` (kapitola 8.6) obsahující oba polygony, v druhém případě bude výsledkem vnější polygon.

Pro operaci vyloučení (XOR) jsou využity metody `sub` a `union` – výsledný `Polygon` nebo `MultiPolygon` se získá sjednocením rozdílu mezi prvním a druhým polygonem a rozdílu mezi druhým a prvním. Výsledek operace vyloučení znázorňuje obrázek 8.4.

8.6 Třída `MultiPolygon`

Samotný polygon pro všechny možné oblasti nestačí, viz obrázky 4.2 a 4.3 v kapitole 4.1. Třída `MultiPolygon` obsahuje množinu objektů třídy `Polygon` z nichž je složená výsledná oblast. Díky tomu je možné reprezentovat oblasti tvořené více polygony (jako např. obec Tišnov na obrázku 4.3) nebo oblasti, které v sobě obsahují díru (jako např. Středočeský kraj na obrázku 4.2). Tato třída reprezentuje oblast tak, že v oblasti leží všechny takové body, které leží v lichém počtu polygonů `MultiPolygon`u – výsledek metody `contains` zavolaná nad objektem třídy `MultiPolygon` je výsledek operace XOR nad návraty metod `contains` všech polygonů tvořící `MultiPolygon`. Např. město Kolín se nachází ve vnějším polygonu oblasti Středočeského kraje, ale ve vnitřním ne – nachází se tedy v jednom polygonu (lichém počtu) a proto spadá do oblasti. Naopak město Praha leží v obou polygonech (v sudém počtu), tudíž do oblasti nespadá.

Třída `MultiPolygon` také obsahuje metodu `optimize`, která mimo volání stejnojmenné metody na všech polygonech také optimalizuje samotnou množinu polygonů. Optimalizuje se např. situace, kdy dva polygony leží částečně přes sebe (jako např. na obrázku 8.4). Pak je možné tyto dva polygony optimalizovat tak, že se rozdělí na dva menší nepřekrývající polygony (znázorněno na stejném obrázku). Také může nastat situace, že `MultiPolygon` obsahuje dva polygony, které se sice nepřekrývají, ale dotýkají se – potom je lze sloučit do jednoho. Některé polygony mohou z `MultiPolygon`u vypadnout úplně, např. když `MultiPolygon` obsahuje dva stejné polygony. V takovém případě je možné oba odstranit, protože když některý bod leží uvnitř jednoho, leží i uvnitř druhého a tedy leží v sudém počtu – takže ve výsledné oblasti bod neleží a stejného výsledku se dosáhne i když v `MultiPolygon`u není ani jeden z těchto dvou polygonů. Tato metoda stejně jako u třídy `Polygon` vrací novou instanci a může se také stát, že se vyřadí z `MultiPolygon`u úplně všechny polygony a v tom případě vrátí tato metoda hodnotu `null`.

Tato třída obsahuje také počítané vlastnosti `leftTop`, `rightBottom`, `size` a `center`, které mají stejné chování, jako u třídy `Polygon`, pouze pracují hromadně nad všemi polygony. Operace `move` a `scale` také volají stejnojmenné metody nad polygony.

Nechybí ani metody `union`, `sub`, `penetration` a `diff`, které mají také stejné chování jako ve třídě `Polygon`, ale ve třídě `MultiPolygon` mají jinou implementaci.

Metoda `union` je implementována pomocí metod `sub` a `diff` tak, že výsledné sjednocení se získá vytvořením rozdílu metodou `sub` a vyloučením tohoto rozdílu a druhého objektu třídy `MultiPolygon` metodou `diff`. Díky implementaci metody `diff` dojde ke sjednocení.

Metoda `sub` provede rozdíl mezi každým polygonem v prvním objektu třídy `MultiPolygon`u a každým polygonem v druhém objektu. Jednotlivé rozdíly spojí vyloučením pomocí metody `diff`. Pokud byl v druhém objektu sudý počet polygonů, provede ještě vyloučení výsledku s prvním `MultiPolygon`em. Výsledkem je rozdíl dvou `MultiPolygon`ů.

Metoda `penetration` provede rozdíl mezi sloučením `MultiPolygon`ů a vyloučením `MultiPolygon`ů. Sloučení vytvoří spojenou oblast původních oblastí a vyloučení vytvoří podobnou oblast z původních oblastí, do které nespadá pouze jejich průnik, který se získá pomocí rozdílu.

Metoda `diff` je díky filozofii třídy `MultiPolygon` velmi jednoduchá. Pouze vytvoří nový objekt třídy `MultiPolygon`, do kterého vloží polygony z obou původních objektů a zavolá nad tímto novým objektem metodu `optimize`.

8.7 Společné metody pro všechny třídy

S výjimkou třídy `Compare` mají všechny třídy některé metody stejné – pouze mají u každé třídy jiné chování. První takovou metodou je metoda `equals`, která porovná dva objekty a určí, zda představují stejný geometrický útvar. U úsečky se neuvažuje orientace (toto chování lze změnit druhým parametrem), u polygonu se neuvažuje, který bod je první a zda jsou body definovány po směru nebo proti směru hodinových ručiček a u třídy `MultiPolygon` se neuvažuje pořadí polygonů.

Všechny třídy také mají definován převod na řetězec, který se využívá při přenosu mezi klientskou a serverovou částí aplikace a také při ukládání do databáze. Pro zpětný převod na objekt slouží statická metoda `fromString`.

Kapitola 9

Klientská část aplikace

Klientská část aplikace je určena pro webové prohlížeče. Proto jsou zde použity jazyky HTML (verze 5) [5], CSS (verze 3) [14] a JavaScript [4]. Klientská část aplikace funguje v rámci pouhé jedné stránky, ve které jsou obsaženy veškeré funkce a data jsou získávána asynchronně pomocí technologie AJAX. Pro zobrazení mapy je využito API[1] mapového portálu Mapy.cz [18].

Tato kapitola popisuje pouze programovou stránku klientské části aplikace. Pro popis aplikace z uživatelského hlediska slouží návod v příloze B.

9.1 Stránka aplikace

Stránka aplikace se nachází v souboru `index.php` a je napsána v jazyce HTML. Obsahuje pouze základní HTML kostru s několika elementy, ve kterých se bude pomocí skriptu dynamicky vytvářet obsah. Design aplikace je zajištěn pomocí kaskádových stylů, jejichž definice se nachází v souboru `style.css`. Tento soubor je připojen v hlavičce HTML. Zde jsou připojeny i soubory se skripty napsané v jazyce JavaScript: knihovna jQuery [17] (slouží pro jednodušší práci s DOM[2]), API portálu Mapy.cz a soubor `js/scripts.js` obsahující vlastní utility vytvořené pro tuto aplikaci. Vlastní logika aplikace se nachází v souboru `js/map.js`, který je připojen až na konci těla HTML, protože předpokládá již existující HTML elementy popsané v tělě stránky.

V souboru `index.php` je také řešeno propojení aplikací, viz kapitola 9.5.

9.2 Utility

Utility pro aplikaci se nachází v souboru `js/scripts.js.php`, který je poskládán z dílčích souborů ve složkách `js/utils`, `js/gui` a `js/polygons`.

V souboru `js/utils/class.js` je popsána funkce `createClass`, která zastupuje definici třídy z třídních jazyků. Nejnovější verze jazyka JavaScript již obsahuje syntaxi pro vytvoření třídy, ale tato syntaxe není v této době podporována staršími verzemi webových prohlížečů, které jsou mezi uživateli stále hodně rozšířené. Takto vytvořené třídy podporují vlastnosti, metody a jednoduchou dědičnost. Dále budu volání této funkce popisovat jako definici třídy.

Soubor `js/gui/DateInput.js` obsahuje definici třídy, která spravuje textové pole pro zadání data. Ošetřuje uživatelský vstup tak, aby nebylo možné zadat neplatné datum.

[1] Application Programming Interface – aplikační programové rozhraní

[2] Document Object Model – objektová reprezentace HTML elementů v jazyce JavaScript

Textové pole podporuje zadání data ve formátu `DD.MM.RRRR` nebo `DD. měsíc RRRR`, přičemž rok je nepovinný – doplní se aktuální rok. Objekt této třídy obsahuje vlastnost `value`, pomocí které lze získat nebo nastavit zadané datum.

Dále v souboru `js/gui/LayoutWindow.js` se nachází definice třídy, která slouží pro zobrazení modálního okna v aplikaci. Toto okno není skutečné okno operačního systému, ale pouze HTML element nastylovaný tak, aby připomínal modální okno. Modální okna jsou v aplikaci řešena tímto způsobem proto, protože vestavěné funkce pro volání modálních oken mají omezené možnosti – nelze je stylovat, není do nich možné vložit libovolný HTML kód apod.

Ve složce `js/polygons` jsou definovány třídy pro práci s polygony implementované v jazyce JavaScript, které jsou popsány v kapitole 8.

9.3 Mapy.cz API

Pro zobrazení mapy a hranic na mapě je v aplikaci využito API mapového portálu Mapy.cz. Toto API dovoluje webovým aplikacém třetích stran zobrazit interaktivní mapu stejnou, jako na webu Mapy.cz. Pomocí API je možné zvolit ovládání mapy, ovládací prvky, vykreslovat na mapě vlastní body nebo geometrii, ovládat mapu pomocí skriptu a další. Pro vývojáře je dostupná dokumentace [19]. V této aplikaci je využito zobrazení mapy se základními ovládacími prvky a možností změnou mapového podkladu. Pro vykreslení území a tras je využita geometrie a pro zobrazení míst a úpravu bodů hranic značky.

9.4 Vlastní kód aplikace

Vlastní kód aplikace se nachází v souboru `js/map.js`, který je také sestavován pomocí PHP, a to z dílčích souborů uložených ve složce `js/map`. Některé z těchto souborů obsahují funkce určené pro uživatele s administrátorským oprávněním. Pokud přihlášený uživatel nemá toto oprávnění, do tohoto souboru se dané dílčí soubory nevkládají.

Soubory určené pro všechny uživatele jsou následující:

`menu.js` – Vkládá do stránky tlačítka menu, které se nachází v pravé horní části aplikace a navazuje na ně dané akce.

`base.js` – Vkládá do stránky mapu a provádí její základní konfiguraci.

`cancel.js` – Definuje „callback“ pro zrušení aktuálně prováděné akce – např. zruší označení základního regionu nebo uživatelského území, odstraňuje editační formulář z bočního panelu apod.

`date.js` – Vkládá do středu horní lišty textové pole s datem. Toto datum představuje aktuální čas platnosti právě zobrazených základních regionů. Změnou hodnoty tohoto pole dojde k znovunačtení dat těchto regionů.

`actualView.js` – Obsahuje funkci pro změnu informace, který typ regionu (obce, okresy, ...) je aktuálně zobrazován. Tato informace se nachází v horní části mapy.

`regions.js` – Obsahuje základní kód pro zobrazení základních regionů – definice vrstev na mapě, detekce oddálení mapy apod.

`markRegions.js` – Zajišťuje zobrazení informací v bočním panelu o aktuálně vybraném základním regionu.

`redrawRegions.js` – Obsahuje funkci pro překreslení aktuálně zobrazených regionů – např. při posunu mapy načte a zobrazí regiony podle aktuálního pohledu mapy.

`removeRegions.js` – Obsahuje funkce pro odstranění zobrazenách regionů na mapě – po posunu mapy, nebo změny zoomu.

`showGroup.js` – Obstarává načtení a zobrazení informací o uživatelském území.

`showPrimitive.js` – Obstarává načtení a zobrazení informací o bodu, trase nebo ploše.

`multiPolygon2SMapGeometry.js` – Obsahuje funkci pro převod třídy `MultiPolygon` (viz kapitola 8.6) na geometrii pro mapové API.

`help.js` – Funkce pro zobrazení návodu k aplikaci.

Soubory, které obstarávají funkce určené pouze pro uživatele s administrátorským oprávněním, jsou tyto:

`hideRegions.js` – Obsahuje funkce pro skrývání a zobrazování základních regionů, které jsou využívány v dalších částech aplikace.

`editRegion.js` – Funkce pro zobrazení formuláře pro editaci základního regionu. Obstarává také odeslání dat na server.

`splitRegion.js` – Obsahuje funkci pro zobrazení formuláře na rozdělení základního regionu na dva nové. Na mapě je možné pomocí myši naklikat dělící křivku, která rozdělí oblast daného regionu na dvě nové.

`joinRegion.js` – Funkce pro zobrazení formuláře pro spojení více základních regionů do jednoho. Na mapě je možné označit regiony, které se spojí dohromady.

`formDateInterval.js` – Obsahuje funkci, která do formuláře vloží políčka pro časovou platnost daného území.

`regionDateLimits.js` – Obsahuje funkci pro načtení minimální a maximální časové platnosti, kterou lze nastavit konkrétnímu základnímu regionu, a také funkci pro kontrolu těchto mezí.

`editGroup.js` – Funkce pro zobrazení editačního formuláře uživatelského území. Na mapě je možné naklikat základní regiony, které budou součástí tohoto území.

`editPrimitive.js` – Funkce pro zobrazení editačního formuláře bodu, trasy nebo plochy. Na mapě je možné naklikat definiční body geometrie dané entity.

`setRegionColor.js` – Obsahuje funkci, která danému regionu vykreslenému na mapě nastaví zadanou barvu..

`selectRegion.js` – Obsahuje funkci pro výběr základního regionu na mapě a jeho uložení – slouží pro propojení s aplikací na zobrazování genealogických dat.

9.5 Propojení aplikací

Aplikace vytvářená v rámci bakalářské práce *Zobrazování genealogických dat* [29] mimo jiné obsahuje farnosti, matriky a obce, ve kterých se nachází. Všechny tyto tři entity mají svá území a aplikace, o které pojednává tato práce, zajišťuje jejich zobrazení na mapě. Obce jsou vázané přímo na základní regiony, území farností a matrik v této aplikaci představují uživatelská území. Zobrazení příslušného území na mapě je řešeno předáním patřičného identifikátoru entity při otevírání stránky aplikace metodou `GET`. Stránku aplikace představuje soubor `index.php`, ten tyto parametry zpracovává a do kódu vkládá příslušný skript v jazyce JavaScript. Tímto způsobem lze i tato území vytvářet nebo editovat, a to tak, že se předá parametr `action` s hodnotou `create`.

V obou aplikacích existují u daných entit hypertextové odkazy, pomocí kterých je možné mezi jednotlivými aplikacemi přepínat.

Kapitola 10

Propojení dat obou aplikací

Databáze vytvořena v rámci bakalářské práce *Zobrazování genealogických dat* [29] obsahuje mimo jiné databázi částí obcí, které jsou součástí farností a farnosti jsou součástí matrik.

Tyto části obce by měly být navázané na příslušné základní regiony v této databázi. Za účelem tohoto propojení byl vytvořen program uložený v souboru `private/obce.php`. Ten projde veškeré části obcí v databázi a pokusí se nalézet v tabulce základních regionů všechny katastrální území (nejlépe odpovídající typ základního regionu části obce) se stejným názvem. Pokud se takový název nalezl právě jeden, je možné provést propojení a program tak učiní. Navíc se kontroluje, zda se daný region nachází na Moravě, protože databáze pro zobrazování genealogických dat obsahuje pouze data z Moravy.

Toto propojení se povedlo zhruba u poloviny částí obcí. Důvodem je fakt, že databáze práce pro zobrazování genealogický dat obsahuje především historická data a moje databáze obsahuje aktuální data z RÚIAN. Proto dnes již některé názvy neodpovídají. Dalším důvod může být, že existuje více obcí se stejným názvem a tudíž je není možné jednoznačně identifikovat a propojit s příslušnými základními regiony. Někde se zase mohl název v průběhu času pozměnit. Navíc je diskutabilní, jak pojem „část obce“ odpovídá pojmu „katastrální území“.

Farnosti se v databázi pro zobrazování genealogických dat skládají z jednotlivých částí obcí. Jejich území lze tedy definovat vytvořením uživatelského území poskládaného z jednotlivých základních regionů odpovídajících částem obce, ze kterých se skládá farnost. Do matriky patří určité farnosti, a tak území matrik lze poskládat podobným způsobem. Tuto akci provádí program uložený v souboru `private/fary-matriky.php`.

Kapitola 11

Navrhy na zlepšení a další využití

Na závěr této práce doplním své návrhy na další vylepšení popsané aplikace a také návrhy na další využití.

11.1 Návrhy na zlepšení

Základní regiony by mohly být rozšířeny o další úroveň – stát. Díky tomu by se mohly definovat základní regiony i v dalších zemích. Nyní neexistuje způsob, jak v rámci rozhraní aplikace definovat základní regiony mimo Českou republiku, protože aplikace podporuje pouze změny v rámci již existujících regionů. Proto by pro toto rozšíření musel v aplikaci vzniknout nástroj pro definování základních regionů na území, na kterém ještě žádný definovaný není. Také by se musel vyřešit problém s rozdílným administrativním dělením v cizině.

Uživatelská území nepodporují změny hranic v průběhu času (časová platnost lze nastavit pouze pro území jako celek) a toto území lze poskládat pouze ze základních regionů. Kdyby se do definice hranic uživatelských území přidala časová platnost a bylo by možné do území zahrnout i plochy (polygony definované nezávisle na základních regionech), přineslo by to nové možnosti.

Aplikace by mohla obsahovat vyhledávání pro vyhledávání území, ať by se už jednalo o základní regiony, uživatelská území, nebo další entity.

11.2 Návrhy dalších využití

Nyní je aplikace propojená s bakalářskou prací *Zobrazování genealogických dat* [29]. Aplikace slouží pro zobrazování farností a matrik z databáze vytvořené v rámci této práce. Aplikace by mohla zobrazovat prakticky libovolná území – panství, hrabství a jiná historická území. Možnosti jsou široké, na mapě se dá zobrazovat pomocí hranic velké množství různých území. Např. při volbách by se dalo pomocí uživatelských území vyznačit oblasti, ve kterých daná politická strana získala nejvíce hlasů.

Aplikace by se mohla rozšířit tak, že by přibyla další úroveň základních regionů – pozemek. Pak by bylo možné aplikaci použít pro zobrazování katastrálních map.

Kapitola 12

Závěr

Nejdříve jsem prostudoval OpenStreet mapy a možnosti, jak z jejich dat získat hranice pro obce. Při tomto studiu jsem se dočetl, že data pro tyto hranice byla získána ze státního registru RÚIAN. Proto jsem si zjistil informace i o tomto zdroji a dospěl jsem k závěru, že je výhodnější získat data odtud.

Vytvořil jsem program, který získal informace a hranice z RÚIAN a uložil je do mé databáze, kterou jsem navrhl. Následně jsem navrhl aplikaci, která má za úkol zobrazovat a editovat tyto hranice a také nad těmito hranice dát možnost definovat vlastní území. Pro účely aplikace jsem rozšířil databázi o nové tabulky a aplikaci jsem implementoval. Pro serverovou část byl použit jazyk PHP a pro klientskou část určenou pro webové prohlížeče jazyky HTML, CSS a JavaScript.

Aplikace byla propojena s bakalářskou prací *Zobrazování genealogických dat* [29], pro kterou zobrazuje území farností a matrik.

Pro aplikaci jsem vytvořil návod a navrhl vylepšení a další využití.

Literatura

[1] *The Apache HTTP Server Project*. 1997-2017, [Online; navštíveno 11.5.2017].
URL `https://httpd.apache.org/`

[2] *OpenStreetMap*. 2004, [Online; navštíveno 26.1.2017].
URL `https://www.openstreetmap.org`

[3] *OpenStreetMap*. 2004, [Online; navštíveno 26.1.2017].
URL `https://www.openstreetmap.org/copyright`

[4] *ECMAScript Language Specification – ECMA-262 Edition 5.1*. 2011, [Online; navštíveno 13.5.2017].
URL `https://www.ecma-international.org/ecma-262/5.1/`

[5] *HTML5*. 2014, [Online; navštíveno 13.5.2017].
URL `https://www.w3.org/TR/html5/`

[6] *RFC 7159 – The JavaScript Object Notation (JSON) Data Interchange Format*. 2014, [Online; navštíveno 11.5.2017].
URL `https://tools.ietf.org/html/rfc7159`

[7] *Elements — OpenStreetMap Wiki*. Srpen 2016, [Online; navštíveno 26.1.2017].
URL `http://wiki.openstreetmap.org/wiki/Elements`

[8] *Planet.osm — OpenStreetMap Wiki*. Prosinec 2016, [Online; navštíveno 26.1.2017].
URL
`http://wiki.openstreetmap.org/wiki/Planet.osm#Country_and_area_extracts`

[9] *Tag:boundary=administrative — OpenStreetMap Wiki*. Listopad 2016, [Online; navštíveno 26.1.2017].
URL `http://wiki.openstreetmap.org/wiki/Tag:boundary%3Dadministrative`

[10] *VDP Výměnný formát*. 2016, [Online; navštíveno 26.1.2017].
URL `http://vdp.cuzk.cz/vdp/ruian/vymennyformat/vyhledej`

[11] *Veřejný dálkový přístup – přístup k datům RÚIAN*. 2016, [Online; navštíveno 26.1.2017].
URL `http://vdp.cuzk.cz`

[12] *Veřejný dálkový přístup (VDP) Uživatelská dokumentace*. 2016, verze 0.5.
URL `http://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/10-Verejny-dalkovy-pristup-informace/Verejny-dalkovy-pristup-informace/VDP_uzivatelska_dokumentace.aspx`

[13] *ČÚZK – RÚIAN*. 2016, [Online; navštíveno 26.1.2017].
URL http://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/RUIAN.aspx

[14] *CSS Current Status – W3C*. 2017, [Online; navštíveno 13.5.2017].
URL https://www.w3.org/standards/techs/css#w3c_all

[15] *Geography Markup Language*. 2017, [Online; navštíveno 30.4.2017].
URL http://www.opengeospatial.org/standards/gml

[16] *Index of /planet*. Leden 2017, [Online; navštíveno 26.1.2017].
URL http://planet.openstreetmap.org/planet/

[17] *jQuery*. 2017, [Online; navštíveno 13.5.2017].
URL http://jquery.com/

[18] *Mapy.cz*. 2017, [Online; navštíveno 13.5.2017].
URL https://mapy.cz/

[19] *Mapy.cz: JsDoc Reference – Seznam tříd*. 2017, [Online; navštíveno 13.5.2017].
URL http://api.mapy.cz/doc/

[20] *MySQL*. 2017, [Online; navštíveno 26.1.2017].
URL http://www.mysql.com

[21] *Nette Framework*. 2017, [Online; navštíveno 11.5.2017].
URL https://nette.org/cs/

[22] *OSM XML -– OpenStreetMap Wiki*. Leden 2017, [Online; navštíveno 26.1.2017].
URL http://wiki.openstreetmap.org/wiki/OSM_XML

[23] *PHP Data Objects*. 2017, [Online; navštíveno 11.5.2017].
URL http://php.net/manual/en/book.pdo.php

[24] *PHP: Hypertext Preprocessor*. 2017, [Online; navštíveno 26.1.2017].
URL http://php.net

[25] *PHP: Sessions*. 2017, [Online; navštíveno 11.5.2017].
URL http://php.net/manual/en/book.session.php

[26] *PHP: spl_autoload_register*. 2017, [Online; navštíveno 11.5.2017].
URL http://php.net/manual/en/function.spl-autoload-register.php

[27] *The DomNode class*. 2017, [Online; navštíveno 30.4.2017].
URL http://php.net/manual/en/class.domnode.php

[28] *The XMLReader class*. 2017, [Online; navštíveno 30.4.2017].
URL http://php.net/manual/en/class.xmlreader.php

[29] VALECKÝ, D.: *Zobrazování genealogických dat*. Brno, 2017, bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D. V přípavě.

[30] ZAMRZLA, J.: *JTSK_Converter*. Listopad 2015, [Online; navštíveno 30.4.2017].
URL https://github.com/josefzamrzla/JTSK_Converter

Přílohy

Seznam příloh

Příloha A

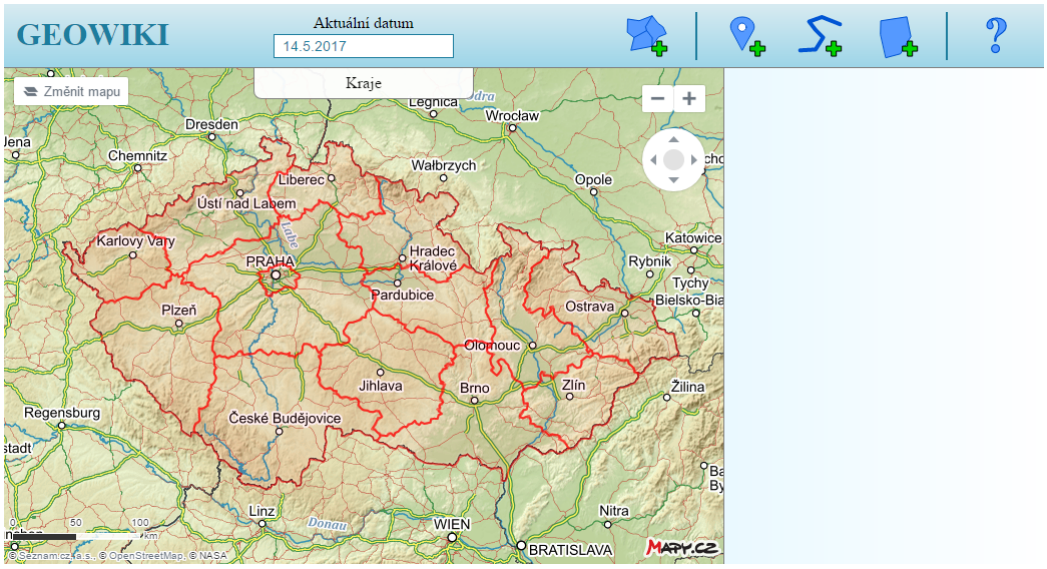
Obsah CD

`src/` – Zdrojové soubory aplikace
`src/private/` – Programy pro jednorázové použití pro převod dat a vytvoření struktury
`db/db.sql` – Databáze aplikace ve formátu SQL
`doc/ibp-xhypes00.pdf` – Technická zpráva ve formátu PDF
`doc/tex/` – Zdrojové soubory pro vytvoření technické zprávy pomocí LaTeXu

Příloha B

Návod k aplikaci

Aplikace je dostupná na adrese `http://perun.fit.vutbr.cz/geowiki/mapa/`. Při otevření aplikace se zobrazí mapa České republiky se zobrazenými hranicemi krajů.



V horní liště se nachází textové pole s datem, pro které jsou na mapě zobrazena data. Toto datum je možné přepsat a hranice na mapě se automaticky překreslí. V pravé části lišty se nachází tlačítka s akcemi. Nepřihlášení uživatelé nebo uživatelé bez administrátorského oprávnění zde vidí pouze tlačítko pro zobrazení návodu. Vpravo od mapy se nachází boční panel, ve kterém se budou zobrazovat informace o aktuálně vybraném území, editační formuláře apod.

B.1 Přihlášení

Přihlášení je nutné provést v aplikaci *Zobrazování genealogických dat* dostupné na adrese `http://perun.fit.vutbr.cz/geowiki/`, kde je možné provést i registraci. Pokud jste přihlášeni v jedné aplikaci, jste zároveň přihlášeni i v druhé. Pokud jste přihlášen jako uživatel s administrátorským oprávněním, můžete editovat, vytvářet nebo mazat území zobrazovaná na mapě.

B.2 Základní regiony

Základní regiony v aplikaci představují základní administrativní členění v České republice. Tyto regiony mají pět úrovní: kraje, okresy, obce, katastrální území a základní sídelní jednotky. V tomto pořadí pro ně platí, že oblast nadřazeného regionu tvoří oblasti podřazených regionů, které danému nadřazenému náleží. Každý region má právě jeden nadřazený – kromě krajů, které nemají žádný. Zobrazení určité úrovně základních regionů je závislé na přiblížení mapy – čím více je mapa přiblížena, tím menší základní regiony jsou zobrazeny. Při kliku na oblast některého základního regionu se v bočním panelu zobrazí informace o tomto regionu. Uživatelé s administrátorským oprávněním zde uvidí i tlačítka pro úpravu, rozdělení regionu a spojení více regionů do jednoho.

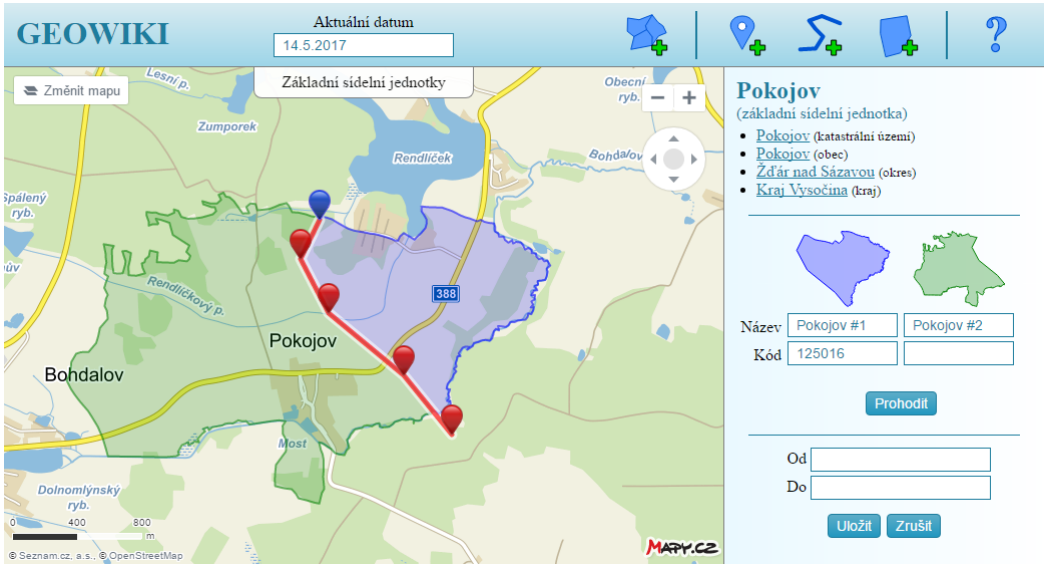
B.2.1 Úprava regionu

Upravit lze název regionu, kód – identifikátor používaný ve státním registru RÚIAN (Registr územní identifikace, adres a nemovitostí) a časová platnost. Při nastavení menší časové platnosti, než která byla před úpravou, se údaje změní pouze v rámci tohoto časového intervalu a údaje před a po tomto intervalu zůstanou nezměněná. Někdy je možné zadat i větší časový interval – ale pouze, pokud lze jednoznačně identifikovat, která verze regionu předchází, respektive následuje – v opačném případě vám toto aplikace nedovolí. Pokud se časová platnost nezmění, znamená to, že není definována a takovému regionu nemůže nic předcházet/následovat po něm. Také lze změnit, kterému nadřazenému regionu region patří – toto je možné provést vybráním příslušného regionu z nabídky.

B.2.2 Rozdělení regionu

Rozdělit lze pouze regiony typu základní sídelní jednotka, a to tak, že rozdělením oblasti tohoto regionu vzniknout dva nové. Jiné typy regionů nelze rozdělit proto, protože obsahují podřazené regiony a rozdělením by u některých mohla nastat situace, že by částečně měly patřit jednomu a částečně druhému nově vzniklému regionu a to nelze, protože podřazený region může mít pouze jeden nadřazený.

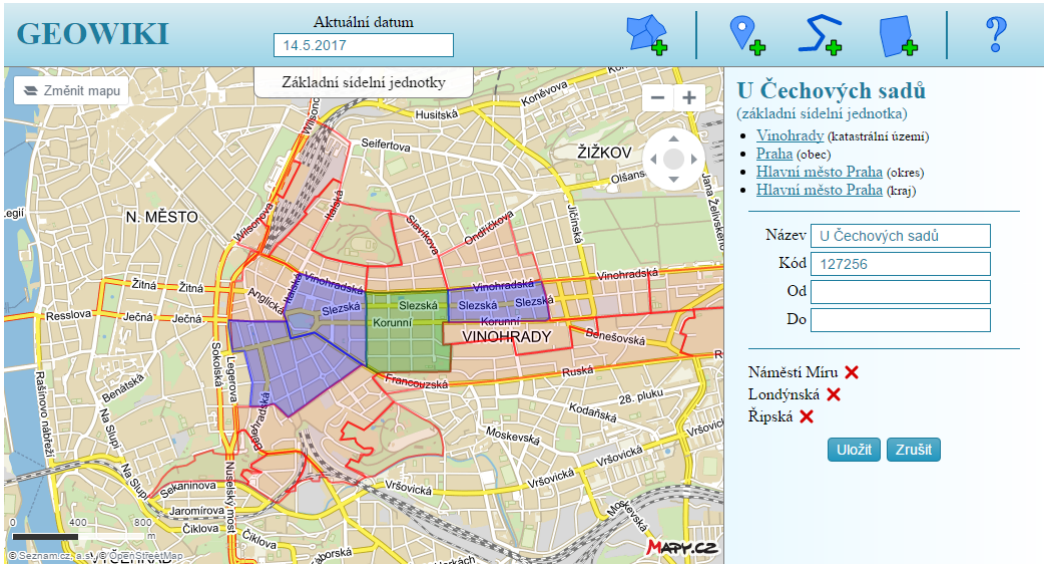
Pro rozdělení je nutné zadat název a kód obou nově vzniklých regionů. Rozdělení oblasti se vytvoří naklikáním dělící křivky na mapě.



Rozdělením oblasti se na mapě zobrazí dvě nové oblasti - modrá a zelená. Tyto oblasti se zobrazí i ve formuláři u zadávání údajů k příslušnému novému regionu. Tlačítko „Prohodit“ prohodí vyplněné údaje mezi těmito regiony. Časová platnost se nastavuje pro oba regiony naráz a také ji lze nevyplnit.

B.2.3 Spojení regionů

Spojit lze regiony na stejné úrovni a to pouze ty, které mají stejný nadřazený region. To je opět z důvodu, aby sloučením regionů nevznikla situace, že by takto součený region měl patřit více nadřazeným.



Spojení oblastí se opět provádí na mapě a to vybráním příslušných regionů. Pokud na region kliknete znovu, opět se jeho vybrání zruší. Přidáné regiony mají modrou barvu, původní zelenou. Všechny modré a zelené oblasti se po uložení spojí do jedné. Vybrané regiony lze odstranit také v bočním panelu, pokud u daného regionu kliknete na křížek.

I u této akce lze upravit údaje o regionu a nastavit časovou platnost.

B.3 Uživatelská území

Uživatelská území mohou uživatelé s administrátorským oprávněním definovat pomocí oblastí základních regionů – lze kombinovat oblasti různých úrovní regionů. Vytvoření nové oblasti lze provést kliknutím na příslušné tlačítko v menu v horní liště. Jednotlivé oblasti se vybírají na mapě stejně jako u spojování základních regionů. Také lze nastavit časová platnost tohoto území.

Tato území se normálně na mapě nezobrazují – ve výchozím stavu aplikace se zobrazují základní regiony. Při označení některého základního regionu se pod informacemi o něm v bočním panelu zobrazí i seznam uživatelských území, jejichž součástí je i vybraný region. Při kliku na některé uživatelské území se na mapě zobrazí oblast tohoto území a v bočním panelu se zobrazí informace o tomto území. Toto mohou provést i uživatelé bez administrátorského oprávnění. Uživatelé s administrátorským oprávněním zde mají navíc možnost toto území upravit, nebo odstranit.

Uživatelská území jsou použita pro zobrazování území farností a matrik v aplikaci *Zobrazování genealogických dat*, ve které je možné při rozkliknutí dané farnosti nebo matriky odkazem přejít do této aplikace, kde se příslušné uživatelské území zobrazí. Pokud farnost nebo matrika doposud nemá definované uživatelské území, lze ho definovat klikem na příslušný odkaz v aplikaci na zobrazování genealogických dat. U takto propojených uživatelských území se v bočním panelu zobrazuje odkaz do aplikace na zobrazování genealogických dat, ve které se zobrazí podrobnosti o příslušné farnosti nebo matrice.

B.4 Body, trasy a plochy

Na mapě mohou uživatelé s administrátorským oprávněním také definovat body, trasy (křivky) a plochy (polygony). Tyto entity jsou definované nezávisle na základních regionech. Vytvořit je lze pomocí tlačítek v horní liště. Na mapě lze naklikat myší jejich geometrii a v bočním panelu vyplnit název, popis a časovou platnost. Vytvořené body, trasy a plochy se také zobrazují pod informacemi o základním regionu, jako uživatelská území, a také je lze rozkliknout pro zobrazení informací o nich a případně je upravit nebo odstranit.