VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO SPRÁVU VIZUALIZACÍ GEOGRAFICKÝCH DAT

INFORMATION SYSTEM FOR MANAGEMENT OF GEOGRAPHICAL DATA VISUALIZATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAN GROSSMANN**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Grossmann Jan, Bc.**
Program: Informační technologie
Obor: Informační systémy
Název: **Informační systém pro správu vizualizací geografických dat**
Information System for Management of Geographical Data Visualizations
Kategorie: Informační systémy

Zadání:

1. Prozkoumejte existující systémy pro tvorbu diagramů umožňující vizualizovat vlastní geografické datové sady. Prostudujte použité typy diagramů a způsob provázání dat s těmito diagramy. Zhodnoťte náročnost tvorby takových diagramů a jejich výslednou použitelnost.
2. Prostudujte principy tvorby informačních systémů, webových uživatelských rozhraní a vizualizací dat. Studujte způsoby uložení vizualizovaných dat.
3. Analyzujte požadavky uživatelů pro snadnou tvorbu mapových diagramů, které mohou být exportovány nebo vloženy do vlastních webových stránek a srovnejte je s možnostmi systémů z bodu 1.
4. Navrhněte informační systém řešící požadavky z bodu 3, který umožní jeho provázání s různými databázovými systémy a dokáže spravovat uživatelské konfigurace vytvořených map. Výsledné mapy bude možné vkládat do vlastních webových prezentací.
5. Navržený informační systém implementujte.
6. Výsledné řešení otestujte.

Literatura:

- Kachlík, J.: Grafická vizualizace geografických dat síťového provozu. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.
- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers/Elsevier, 2010, ISBN: 978-0-12-375030-3.
- Few, S.: *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly, 2006, ISBN: 978-059-6100-162.
- Leaflet: *Leaflet API reference* [online]. 2019 [cit. 2020-09-16]. Dostupné z: https://leafletjs.com/reference-1.7.1.html

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hynek Jiří, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 22. října 2020

Abstrakt

Cílem práce je vytvořit informační systém pro správu vizualizací geografických dat. Hlavní myšlenkou je umožnit uživatelům tvorbu vizualizací z jejich vlastních geografických dat, které mohou buď nahrát do systému pomocí souborů, nebo získat připojením jejich vlastní databáze. Uživatelská databáze poslouží jako zdroj dat a bude schopna reflektovat změny v reálném čase. Výsledkem práce bude nový informační systém, který poslouží jako styčný bod mezi uživateli, geografickými daty a vizualizacemi těchto dat.

Abstract

The goal of this this is to create an information system for the visualization of geographical data. The main idea is to allow users to create visualizations with their own geographical data, which they can either import from files or directly attach their own database system as a source of data and make use of the data in real-time. The result will be a new web information system that will act as a point of contact between users, geographical data, and visualizations.

Klíčová slova

informační systém, geovizualizace, geografická data, databázové systémy, JavaScript, React, MongoDB

Keywords

information system, geovisualization, geographical data, database systems, JavaScript, React, MongoDB

Citace

GROSSMANN, Jan. *Informační systém pro správu vizualizací geografických dat.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek, Ph.D.

Informační systém pro správu vizualizací geografických dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jiřího Hynka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Grossmann
17. května 2021

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Jiřímu Hynkovi, Ph.D. za odborné vedení, čas, ochotu a cenné rady, ale také své rodině a přátelům za podporu při tvorbě této práce.

Obsah

Kapitola 1

Úvod

V dnešní době již není problém automaticky zaznamenávat a uchovávat velké množství dat. Tyto data se mohou svými atributy pojit například k určité oblasti nebo místu na zeměkouli, ať už se jedná o světadíly, státy, nebo konkrétní místa zadané pomocí souřadnic, čímž se z nich stanou data geografická. Tyto data mohou být uchovávána buď formou zápisu v textových souborech v čitelné formě, ovšem mnohem častěji se data ukládají do rozsáhlých databází, například ve formě tabulek. Tento způsob s sebou ale nese problém opětovného porozumění informacím, kdy sice každý jedinec pochopí jednoduchý příklad záznamu maxima denních teplot v Africe, jenže při použití souřadnic konkrétního místa na Sahaře bude většina lidí bezradná. Právě to řeší geografické informační systémy pomocí vizuální interpretace geografických dat, kdy jsou data nanesena přímo na mapové podklady a umožňují i určitou míru interakce.

Přestože existuje mnoho služeb a aplikací, které umožňují uživateli spravovat mapové podklady, eventuálně přidávat vlastní body zájmů do mapy, tím dosáhnout určitého stupně vizualizace dat a takto obohacenou mapu poté sdílet, jen málo z nich je natolik uživatelsky přívětivých, že práci s nimi zvládne i běžný počítačový uživatel. Zároveň se již nějaký čas dostávají do popředí aplikace napsané pomocí webových technologií, které nezatěžují uživatele nutností stahovat instalační soubor a nainstalovat aplikaci lokálně. Dále umožňují synchronizaci dat mezi zařízeními a jsou nezávislé na platformě. Právě z těchto důvodů se množství aplikací přesunulo na cloudovou platformu a část z nich tvoří i informační systémy. Může se jednat o aplikace přímo vyrobené na zakázku sloužící k danému účelu, jako například vizualizace výsledků voleb (obarvení okresů podle strany s majoritním podílem) či celosvětový přehled aktuální situace s pandemií COVID-19. Rovněž se může jednat o aplikace sloužící k vizualizaci bodů či trasy na mapě, případně aplikace blížící se cíli této diplomové práce, tedy přehledné a srozumitelné zobrazení včetně správy velkého množství dat s geografickým kontextem.

1.1 Cíl práce

Cílem práce bude vytvořit informační systém nad stávající webovou aplikací Geovisto[1]. Aplikace Geovisto již zvládá po importu všech potřebných dat a souborů zobrazovat a interpretovat surová data v přehledné a interaktivní formě, ovšem zatím nedisponuje prostorem, kde by se tyto data a konfigurace mohly uchovávat. Právě proto vzniká tento informační

[1] https://github.com/geovisto

systém, který bude sloužit jako *wrapper*[2] pro usnadnění práce s aplikací. Informační systém se bude skládat ze dvou částí – frontend a backend. Backendová část bude komunikovat s vlastním databázovým systémem a zaměří se na uchování, zpracování a poskytování datových sad, konfigurací, podpůrných informací, tedy všeho, co se týká uživatelských profilů a konfigurace systému samotného. Konfigurace a datové sady bude možno spravovat skrze intuitivní a přívětivé uživatelské rozhraní ve frontendové části (část informačního systému zobrazována v prohlížeči uživatele). Hlavní důraz bude kladen na poskytnutí aplikace Geovisto i úplným začátečníkům bez jakékoliv znalosti programování za pomocí pár kliknutí. Nebude se tedy jednat o pouhé API (zkratka anglického *Application Programming Interface*[3]), ale o plnohodnotný nástroj. Vytvoření a editace datových sad bude probíhat buď textovou formu přímo v frontendové části, nebo importem souborů různých formátů – například CSV, XML nebo JSON. Další způsob importu dat bude probíhat připojením k vybraným databázovým systémům, se kterými se backendová část naučí zacházet a data v nich uložená zpracovat do přijatelné formy. Tento informační systém bude styčným bodem mezi uživateli, kteří mají požadavky na vizualizaci, různými formáty dat z rozličných zdrojů a funkcemi poskytovanými aplikací Geovisto.

Poslední, ale zároveň velice důležitou funkcionalitou informačního systému, bude možnost sdílení vytvořených vizualizací z informačního systému i s okolním světem, například za pomocí přímého odkazu nebo použitím *embedded*[4] elementů jednoduše vložitelných do cizích webových stránek.

1.2 Obsah práce

V druhé kapitole popisuji, co to jsou geografická data, jakými objekty mohou být reprezentovány, jakým způsobem jsou tyto data uložená a v jakých typech formátů. To právě z důvodu sdílení mezi na sobě nezávislými platformami. Dále představuji, jak se geografická data obsahující geografickou informaci propojí s určitou lokací nebo místem na mapě. V poslední řadě pak, jak lze geografická data vizualizovat v podobě grafů.

Ve třetí kapitole zkoumám stávající možnosti vizualizace geografických dat, tedy řešení problému z kapitoly 2, za použití různě komplexních systémů, od knihoven v podobě API až po geografické informační systémy. Zajímavý pro tuto práci je hlavně pohled na jednoduchost práce s nástrojem, import dat, vytvoření vizualizace a následně možnost použití vizualizace na vlastních webových stránkách nebo sdílení s kolegy.

Čtvrtou kapitolou pronikám do oblasti geografických informačních systémů, zkoumám možnosti použitelných technologií, jak na frontendové straně, tak i na straně backendu a databáze. Dále představuji možnosti, jaké databázové systémy lze přímo použít pro importování již stávajících dat a jaké nástroje lze k tomuto účelu použít.

V páté kapitole se zaměřím na analýzu problematiky vizualizace geografických dat, jaké jsou běžné požadavky, co uživatel od vizualizace očekává, jak toho může docílit a co následně s vytvořenou vizualizací.

V šesté kapitole popisuji návrh geografického informačního systému a popíšu architekturu včetně návrhu klientské, serverové i databázové části. Dále osvětlím jak spolu budou tyto celky komunikovat, jakým způsobem se budou ukládat interní data, s jakými formáty

[2] *Wrapper*, tedy jakýsi obalovač, který v jádru poskytuje funkcionalitu původní aplikace Geovisto, ale přidává další pomocné funkce.

[3] https://developer.mozilla.org/en-US/docs/Glossary/API

[4] *Embedded elementy* rozšiřují HTML dokument pomocí částí načtených z dalších zdrojů nebo HTML prezentací

a typy databází si bude informační systém rozumět a jakým způsobem bude do tohoto systému zakomponována původní aplikace Geovisto.

Pomocí sedmé kapitoly již popíši konkrétní implementační detaily vytvářeného informačního systému. Proniknu hlouběji do problematiky a zmíním zajímavé způsoby použité implementace.

V osmé kapitole se zabývám testováním vyvinutého produktu. Zkoumám jaké chyby obsahuje a zda produkt splňuje očekávané požadavky na funkčnost.

Kapitola 2

Geografická data a jejich vizualizace

Tato kapitola pojednává o problematice geografických dat a referencování těchto dat včetně výčtu objektů, ze kterých se tyto data mohou skládat. Pro dostatečné porozumění je následně nutné vysvětlit, jakým způsobem probíhá provázání dat s geografickými objekty a jak lze tyto data ukládat pro jejich znovupoužití. V poslední části kapitoly popíšu několik způsobů vizualizací dat v podobě tématických map.

2.1 Geografická data

Geografické datové sady jsou kolekce obsahující geoprostorová data (*geospatial data*). Jsou to taková data, ve kterých aspoň jedna z datových dimenzí referuje geografickou polohu [21]. To, co odlišuje geoprostorová data od ostatních prostorových dat, je, že jsou absolutně nebo relativně umístěna na povrchu naší planety nebo jsou georeferencovány (vysvětleno níže). To znamená, že každý záznam z datové sady je spjat s geografickou informací uloženou například ve formě souřadnic, adresy, názvu města nebo poštovního směrovacího čísla. Dále existuje vícero způsobů, jak definovat souřadnicový systém a pozici z něj následně převést do libovolného počtu dalších souřadnicových systémů, například k vytvoření mapového zobrazení.

2.1.1 Georeferencování

Pro pojem georeferencování existují různé definice. Například Sommer a Wade definují georeferencování jako „sladění geografických dat se známým souřadnicovým systémem tak, aby šlo data zobrazit, dotazovat a analyzovat s ostatními geografickými daty.“ [33].

Případně existuje i širší pohled na doménu georeferencování, skládající se ze tří hlavních typů pozorovaných dat.

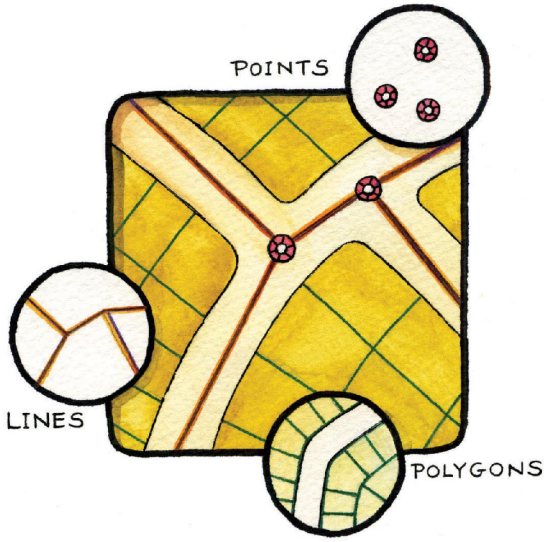
- Geometrické informace popisují geometrické vlastnosti objektu, například rozložení a tvar segmentu silnice.
- Topologické informace se zabývají vlastnostmi, které jsou zachovány pod kontinuálními deformacemi předmětů. V kontextu georeferencí bývá struktura grafu vyvolaná určitými propojeními či sítěmi geografických objektů, jako jsou silnice nebo řeky.

- Sémantické informace zahrnující různé druhy sémantických vlastnosti, které souvisí se zeměpisnou polohou, například název místa, řeky nebo silnice.

Běžné metody georeferencování zohledňují aspoň jeden z těchto typů, ale často tyto informace kombinují k dosažení jednoznačné identifikace geografického objektu [34].

2.1.2 Geografické objekty

V geografických informačních systémech existují tři základní typy objektů – body (*points*), úsečky (*lines*) a mnohoúhelníky (*polygons*). Znázorněny jsou na obrázku 2.1. S využitím těchto základních stavebních kamenů lze pak skládat pokročilejší geografické struktury.



Obrázek 2.1: Body, úsečky a mnohoúhelníky[1]

Body

Body jsou objekty s nulovou dimenzí, což znamená že se jedná o bezrozměrný geometrický útvar. Body obsahují pouze jeden pár souřadnic. Obvykle se používají k modelování singulárních diskrétních prvků, jako jsou studny, sloupy elektrického napětí, body zájmů a podobně. Body se dají dále dělit na uzly a vrcholy, potom se uzlem nazývá průsečík dvou úseček/polygonů reprezentující běžený pár souřadnic (x, y), zatímco vrcholy jsou označovány jako ohyby podél lomené čáry nebo polygonu [31].

Úsečky

Body lze prostorově propojit a vytvořit tak složitější prvky. Úsečky jsou jednorozměrné prvky složené z dvou krajních bodů a obsahují vlastnost v podobě délky. Propojením více bodů za sebou dostáváme lomenou čáru. Úsečky a lomené čáry se používají k reprezentaci lineárních prvků, jako jsou silnice, potoky, zlomy, hranice a podobně [31].

[1] https://saylordotorg.github.io/text_essentials-of-geographic-information-systems/s08-02-vector-data-models.html

Mnohoúhelníky

Mnohoúhelníky, neboli polygony, jsou dvourozměrné prvky vytvořené spojením několika úseček, které se uspořádají do podoby smyčky tak, aby vznikl jeden uzavřený element, který obsahuje vlastnosti plochy a obvodu. V případě mnohoúhelníků je první dvojice souřadnic (x, y) na prvním místě v úsečce stejná jako dvojice souřadnic (x, y) na posledním místě v poslední úsečce. Využití mnohoúhelníků lze najít při reprezentaci hranic města, geologických útvarů, jezer, vegetačních formací atd. Mnohoúhelníky se také mohou nazývat oblastmi [31].

2.2 Formáty geografických dat

Při použití geografických dat různých formátů z různých datových zdrojů nastává mnoho problémů, a proto je potřeba jednotlivé formáty prozkoumat a posoudit jejich charakteristiky. Může se například jednat o tématický obsah dat, kdy v datech nejsou obsažené všechny potřebné položky, užití rozdílných souřadnicových systémů či rozdílné měřítko a přesnost dat.

2.2.1 GeoJSON

GeoJSON je formát určený pro výměnu geoprostorových dat založený na formátu JSON[2]. Definuje několik vlastních typů objektů JSON. Dále pak způsoby, jakým jsou tyto objekty kombinovány tak, aby správně reprezentovaly údaje o geografických prvcích, jejich rysech i prostorových vlastnostech. GeoJSON využívá geografický souřadnicový referenční systém World Geodetic System 1984 s jednotkami desetinných stupňů [7]. GeoJSON je podporován množstvím mapovacích nástrojů a geografických informačních systémů.

Základním kamenem bývá kolekce prvků (tzv. *feature*). Mezi tyto prvky patří body (*Point*), lomené čáry (*LineString*), mnohoúhelníky (*Polygon*) a vícedílné kolekce těchto typů, jako *MultiPoint*, *MultiLineString*, *MultiPolygon* a *GeometryCollection*. Prvky formátu GeoJSON nemusí představovat pouze fyzické entity našeho světa, jako jsou třeba budovy či jiné objekty na které si můžeme reálně sáhnout. Například navigační nebo telekomunikační aplikace mohou popisovat míru pokrytí svých služeb pomocí elementů formátu GeoJSON. Jednoduchou ukázku formátu GeoJSON lze vidět v příkladu 2.1.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Výpis 2.1: Ukázka dat ve formátu GeoJSON.

Jedním z potomků GeoJSONu je TopoJSON, rozšíření formátu GeoJSON, které kóduje geoprostorovou topologii a obvykle poskytuje i menší velikosti souborů.

[2] https://www.json.org

2.2.2 KML

KML je druh formátu XML zaměřený na geografickou vizualizaci, včetně anotací map a obrázků. Geografická vizualizace zahrnuje nejen prezentaci grafických dat na ploše zeměkoule, ale i navigování uživatele ve smyslu, kam jít nebo kde hledat [25]. KML používá běžné typy XML, například *boolean, string, double, float* a *int*. Kromě toho definuje řadu vlastních pokročilých typů jako jsou *dateTime*, *angle90 / angle180 / angle360 color*, *shapeEnum* a podobně. Příklad zápisu formátu KML lze vidět ve výpisu 2.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Výpis 2.2: Příklad uložení dat ve formátu KML.

2.2.3 Shapefile

Shapefile je digitální vektorový formát pro uložení geometrického umístění a souvisejících atributů. Tento formát samotný – ukládaný pomocí souborů s příponou *.shp*, postrádá prostor pro uchování topologických informací. Potřebuje tak další podpůrné soubory mapující atributy prvků na grafické elementy – soubory *.shx* a *.dbf*. Formát Shapefile ukládá data jako jsou primitivní geometrické tvary, body, úsečky a mnohoúhelníky. Tyto tvary pak po spárování s datovými atributy vytvářejí reprezentaci geografických dat.

2.3 Provázání geografických objektů s daty

Jednotlivá data je pro potřeby vizualizace na mapě nutné spárovat s určitou lokací či místem na zemském povrchu. Docházet k tomu může například pomocí zeměpisných souřadnic World Geodetic System 1984, adresy, nebo použitím geokódů Geohash, Open Location Code, eventuálně Alphacode (viz. níže). Vedle volně použitelných řešení, které byly nyní zmíněné, existují i technologie opatřené patentem, jako například Mapcode a What3Words.

Geokód je kód, který představuje geografickou entitu (lokaci nebo objekt) [28]. Někdy může být tento termín použit v širším smyslu: popis sousedství, lokality, případně demografických rysů, jako jsou etnické složení nebo průměrný příjem či úroveň vzdělání skupiny obyvatel, jak je tomu užíváno v marketingu.

2.3.1 Adresa

Jednoduchý a snadno srozumitelný způsob, jak čemukoliv přiřadit geografickou komponentu, je například označit danou věc poštovní adresou. Podle §29 písm. h) Zákona č. 111/2009 Sb., adresou může být brána kombinace názvu okresu, názvu obce, názvu městské části nebo městského obvodu, názvu části obce nebo v případě hlavního města Prahy

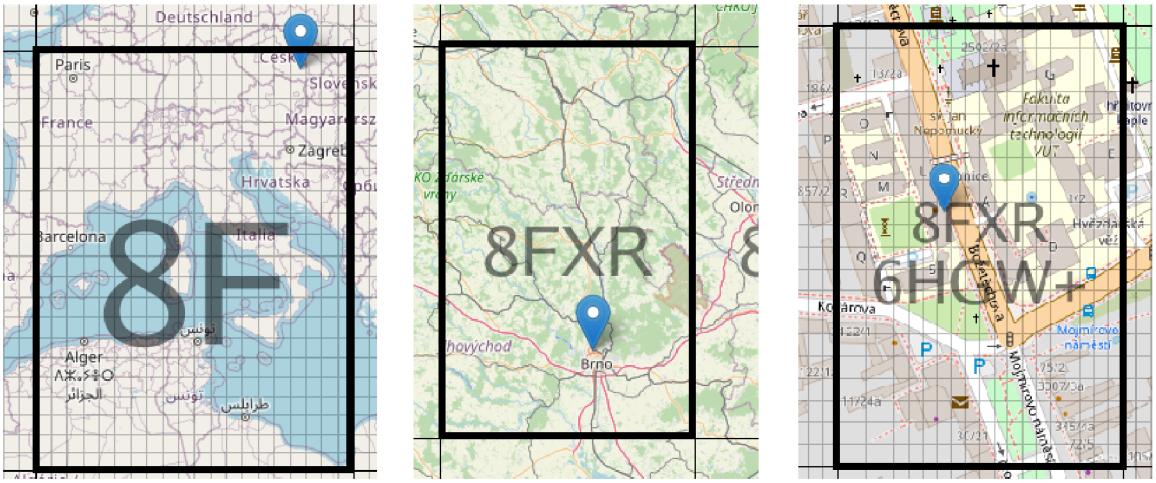
názvu katastrálního území, čísla popisného nebo evidenčního, názvu ulice a čísla orientačního a dále zvláštních údajů pro doručování prostřednictvím poštovních služeb, která jednoznačně určuje adresní místo [29].

2.3.2 Open Location Code

Technologie Open Location Code[3] neboli *pluskód* (dále jen kód) je navržena tak, aby vytvářela kódy, které lze použít jako náhradu za adresy ulic, zejména v místech, kde budovy nejsou očíslovány nebo ulice nejsou pojmenovány. Tyto kódy představují oblast, nikoliv bod. S každým přidáním jednoho z 20 alfanumerických znaků do řetězce kódu, dochází ke zmenšení představované oblasti a tak i k upřesnění pozice (demonstrováno na obrázku 2.2).

Kódy, které si jsou podobné, reprezentující oblasti umístěny blíže k sobě než kódy, které se významným způsobem liší. Algoritmus užívaný pluskódy je veřejně dostupný a lze jej použít bez omezení [10].

Jako příklad lokace zapsané v Open Location Code lze například uvést budovu Fakulty informačních technologií Vysokého učení technického v Brně, na adrese *8FXR6HGW+H9*. První čtyři znaky – 8FXR znázorňují oblast velikostí odpovídající zhruba polovině jihomoravského kraje. Druhé čtyři znaky a část za symbolem plus – 6HGW+H9 pak společně upřesňují polohu v rámci vybrané oblasti.

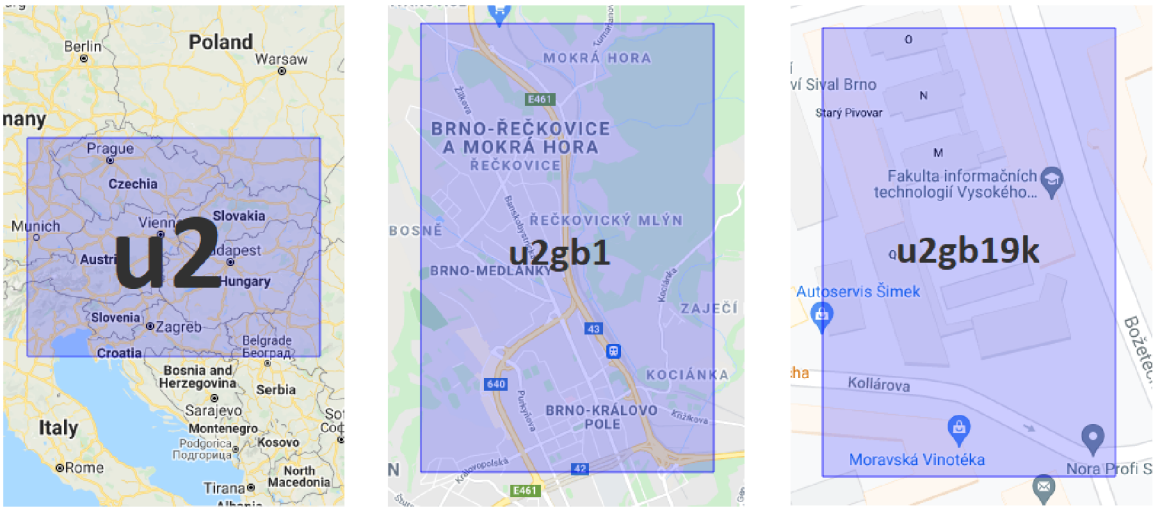


Obrázek 2.2: Znázornění využití mřížky pluskódu při lokalizování vybrané oblasti.

2.3.3 Geohash

Geohashing je metoda geokódování používaná ke kódování zeměpisných souřadnic (zeměpisné šířky a délky) do krátkého řetězce číslic a písmen vymezujících oblast na mapě. Tato oblast se nazývá buňka a zahrnuje různou úroveň přesnosti umístění, která je určená počtem znaků v řetězci. Oblasti na zeměkouli jsou děleny na podoblasti pomocí Mortonovy Z-křivky, každá oblast je zastoupena jedním znakem. Po výběru dané oblasti lze polohu upřesnit rozdělením oblasti na další podoblasti a výběrem přidat další znak do řetězce Geohashe [12]. Zápis stejné lokace jako v případě pluskódu by v okamžiku použití metody geohashe vypadal následovně – *u2gb19kv*. Jak se k výslednému kódu dospělo lze pochopit z obrázku 2.3.

[3] https://maps.google.com/pluscodes/



Obrázek 2.3: Zvyšování úrovně přesnosti kódování pomocí metody Geohash.

2.3.4 ISO 3166-1

Účelem standardu ISO 3166 [2] je definovat mezinárodně uznávané kódy písmen a/nebo čísel, které můžeme použít při odkazování se na země a jejich podčásti. Nedefinuje však názvy zemí – tyto informace pocházejí ze zdrojů Organizace spojených národů.

Používání kódů šetří čas a předchází chybám, protože namísto použití názvu země, který se může lišit v závislosti na používaném jazyce, můžeme použít kombinaci písmen a/nebo čísel, kterým porozumí celý svět.

Alpha-2

Kódy ISO 3166-1 alpha-2 jsou dvoupísmenné kódy představující státy, teritoria, závislá území a oblasti geografického zájmu definované organizací ISO[4]. Jsou nejpoužívanějšími z kódů zemí publikovaných ISO a jsou nejvíce známé například jako domény nejvyšší úrovně internetového kódu země. Kódové označení České republiky pomocí ISO 3166-1 alpha-2 je **CZ** a například Spolkové republiky Německo je **DE**.

Alpha-3

Kódy ISO 3166-1 alpha-3 jsou třípísmenné kódy se stejným zaměřením jako alpha-2 vydané stejnou organizací. Umožňují však lepší vizuální asociaci mezi kódy a názvy států než dvoupísmenné alpha-2 kódy. Poprvé byly zahrnuty jako součást normy ISO 3166 ve stejném roce jako alpha-2. Kódové označení České republiky pomocí ISO 3166-1 alpha-3 je **CZE** a například Spolkové republiky Německo je **DEU**.

2.4 Typy vizualizací

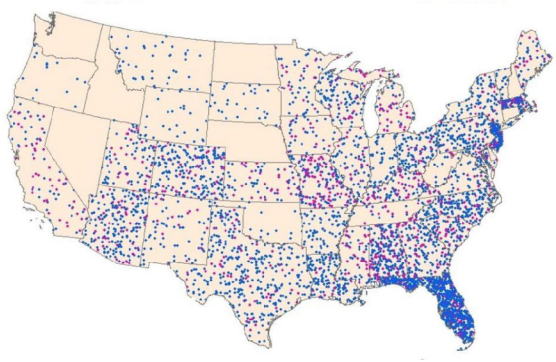
Pro vizualizaci geografických dat se ustálily určité typy tématických map, zobrazující grafické prvky – dříve zmíněné body, mnohoúhelníky apod., nad daným mapovým podkladem.

[4]International Organization for Standardization

Tématické mapy informují o skutečnostech, které jsou předmětem obsahu jednotlivých mapových děl. Zobrazený obsah použitého mapového podkladu umožňuje lokalizaci jednotlivých prvků tématicky zaměřeného obsahu [15]. Podle druhu použití se mohou lišit jak tvarem zobrazeného grafického prvku, tak i barvou. Hodnotu lze poté rozlišit například pomocí velikosti grafického prvku, intenzity barevného odstínu nebo kombinací obou těchto přístupů.

2.4.1 Metoda teček

Metoda teček (z anglického *dot map*, obrázek 2.4) je metoda kartografického znázornění pro vizualizaci diskrétních absolutních hodnot vůči jejich prostorovému rozložení. Tečky stejné velikosti se používají pro reprezentaci stejných hodnot. Pro tento účel se hodí využití geografického objektu typu bod (*point*). Podle hodnoty tečky se k zobrazení datové hodnoty oblasti používá určitý počet teček. Tyto tečky pak obvykle tvoří shluky. Hodnotu dat je třeba zaokrouhlit na násobek hodnoty reprezentované tečkou. Následně je možné zhruba určit hodnotu vizualizovaných dat spočítáním teček a vynásobením počtu hodnotou tečky. Protože existuje mnoho parametrů – velikost bodu, hodnota bodu, měřítko mapy – které je třeba vzít v úvahu při navrhování mapy bodů, je manuální způsob této vizualizace velmi složitý a časově náročný [16].



Obrázek 2.4: Příklad tématické mapy typu mapa teček.[5]

2.4.2 Mapa značek

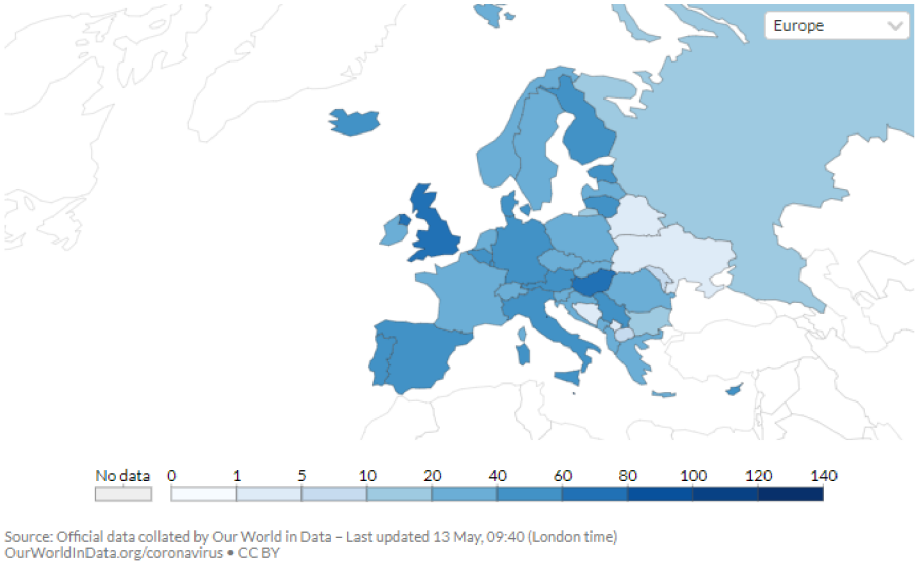
Vyznačením bodu na mapě (anglicky *marker*) lze vizuálním způsobem odkázat na konkrétní geografickou souřadnici [14]. Jako grafická reprezentace se většinou používá kruhová značka, ovšem lze použít i jiné, jako třeba křížek nebo různé speciální znaky. Mapa značek je v mnoha ohledech podobná metodě teček, dá se tak označit za vylepšení této metody. V okamžiku kdy dochází k zobrazení velkého počtu značek na malém prostoru, například vlivem velkého oddálení mapy oblasti a hrozí vzájemné překrývání značek, lze využít tzv. shlukování, tedy vznik speciální značky, která reprezentuje shluk dvou a více standardních značek. Poté musí dojít k vygenerování značky shluku a také k přejmutí vlastností a hodnot všech zastoupených značek [20].

[5] https://twitter.com/gijn/status/1139504012628299777/photo/1

2.4.3 Kartogram

Kartogram (z anglického *choropleth map*, obrázek 2.5) je typ tématické mapy, ve které je soubor předdefinovaných oblastí vyznačen určitou barvou s intenzitou danou pomocí hodnoty statistické proměnné. Tato proměnná představuje souhrn geografických charakteristik v dané oblasti. Jedná se o kvantitativní rozlišení jevů, základem je takzvaný kartografický areál. To je oblast, ve které se sleduje daný jev [1]. Pomocí kartogramu lze snadno vizualizovat, jak se proměnná mění v různých geografických oblastech. Pro takové znázornění se sestaví sledovaná oblast z množství polygonů a každému z nich se přiřadí na sobě nezávislé vizuální vlastnosti. Existují rozdílné druhy kartogramů:

- jednoduchý kartogram,
- složený kartogram,
- strukturní kartogram,
- síťový kartogram,
- objemový kartogram.



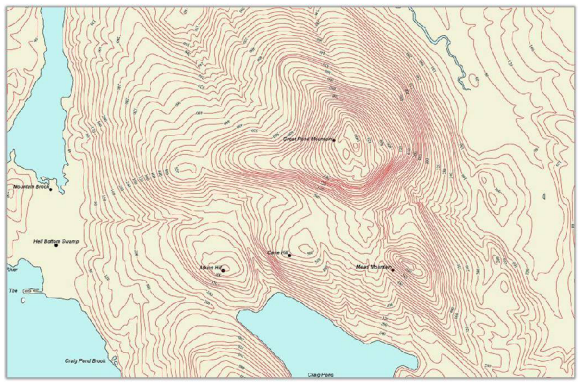
Obrázek 2.5: Příklad tématické mapy typu kartogram.[6]

2.4.4 Isopleth

Mapa typu isopleth (obrázek 2.6) je další z forem geografické vizualizace. Třetí rozměr v podobě hodnoty dat může být propojen s okolím například i pomocí barev, kdy přiřazený odstín barvy znázorňuje násobky daného intervalu. Hlavním rysem v Isopleth mapách je izolinie (anglicky *contour line*), tedy liniová mapová značka spojující místa se stejnými hodnotami dané veličiny. Izolinie lze pro potřeby vizualizace sestavit pomocí úseček. Izolinie se nemohou křížit a jejich vzdálenosti jsou nepřímo úměrné gradientu daného prvku [35].

[6] https://ourworldindata.org/covid-vaccinations

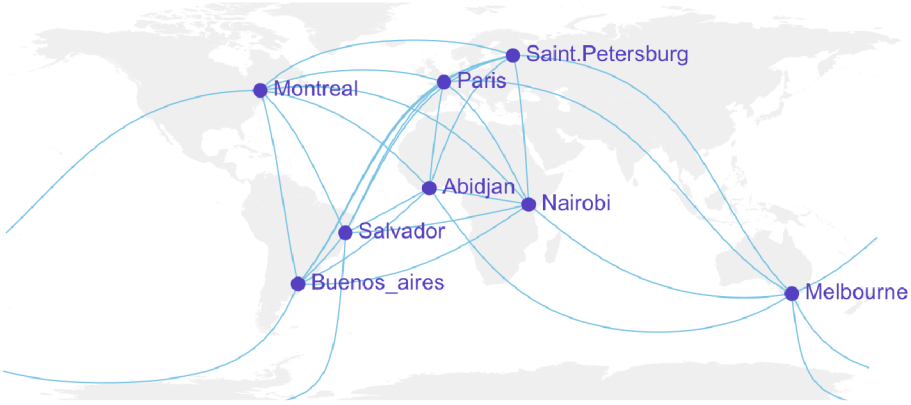
Nejznámějším typem izolinie je vrstevnice. Isopleth mapa tak nalezne použití například při mapování výšek povrchu nebo zobrazení atmosférického tlaku. Tato geovizualizace vychází z kartogramu, ale na rozdíl od něj není hodnota mapována do předdefinovaných polygonů, nýbrž do oblastí, jejichž souřadnice jsou přímo součástí vstupních dat [20].



Obrázek 2.6: Příklad tématické mapy typu isopleth.[7]

2.4.5 Mapa spojení

Mapa spojení (anglicky *connection map*, obrázek 2.7) je velice jednoduchý ale účinný druh tématické mapy. V našem případě zobrazuje hrany znázorňující vztahy mezi několika body nebo geoprostorovými oblastmi na mapě. Spojení bývá většinou reprezentováno ortodromou. Ortodroma je geodetická křivka nanesena na kulové ploše, která vyjadřuje tzv. ortodromickou vzdálenost (nejkratší vzdálenost dvou bodů na ploše). Mapa spojení je také užitečná v případě hledání prostorových vzorů prostřednictvím distribuce jednotlivých spojení na mapě, nebo pro sledování koncentrace spojení v různých oblastech.



Obrázek 2.7: Příklad tématické mapy typu mapa spojení.[8]

[7] https://en.wikipedia.org/wiki/Contour_line#/media/File:Cntr-map-1.jpg
[8] https://www.r-graph-gallery.com/how-to-draw-connecting-routes-on-map-with-r-and-great-circles.html

Kapitola 3

Nástroje pro vizualizaci geografických dat

Tato kapitola představuje již existující nástroje umožňující vizualizaci dat nad mapovými podklady. V této oblasti již existuje množství nástrojů, které se zaměřením blíží i vzdalují cíli této práce. Pro potřeby prosté vizualizace dat a prezentace výsledků na vlastní webové stránce již slouží velké množství aplikačních rozhraní. Mezi nejznámější v České Republice určitě patří rozhraní od Mapy.cz nebo od Google Maps. Co se týče komplexnějších řešení i se správou dat, musíme se poohlédnout po geografických informačních sytémech, neboli GIS. Ty dávají uživateli navíc i možnost importovat a uchovávat data na straně informačního sytému, ale nezřídka se jedná o *enterprise* řešení větších společností, kde je používání podmíněné zaplacením licence či měsíčními poplatky.

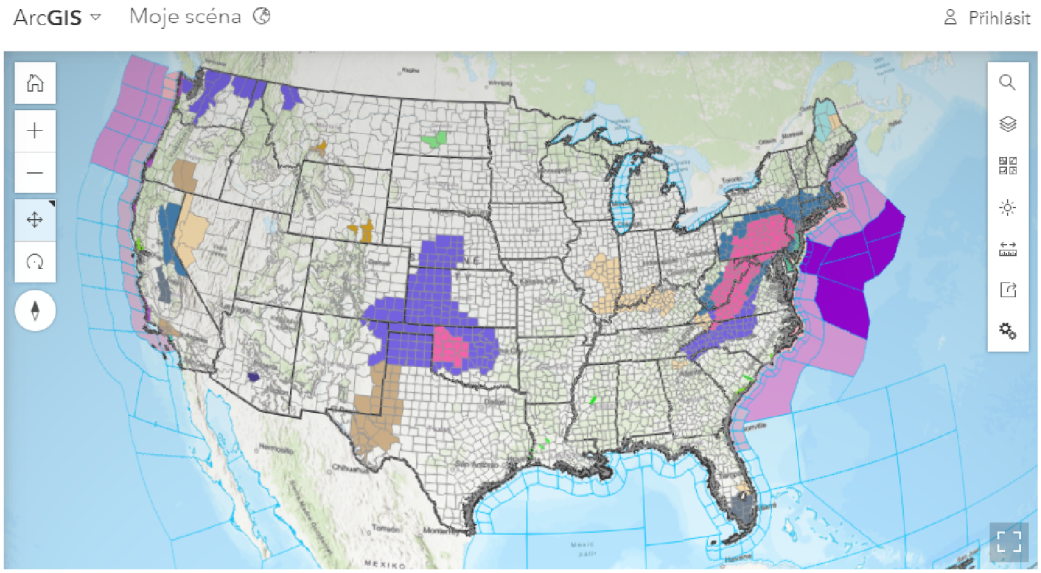
3.1 ArcGIS Online[1]

Je cloudová (*software-as-a-service* – softwarové řešení pro připojení k aplikacím a používání jich přes internet pomocí webového prohlížeče [24]) aplikace vycházející z původně desktopové aplikace ArcGIS[2]. Aplikace ArcGIS Online umožňuje vytvoření určitých relací mezi lokacemi, lidmi nebo libovolnými daty za použití interaktivních map. Kromě možnosti práce s SDK či API poskytuje ArcGIS Online přehledné webové rozhraní pro konfiguraci a správu map. Načtení dat může proběhnout z velkého množství formátů jako například KML, GeoJSON nebo různé tabulkové formáty. Interaktivní mapa je dokonce i schopna průběžné aktualizace dat z živých streamů. Právě množství živých streamů, které jsou již připravené k přímému použití, může být lákadlem k použití systému. Nenutí tak uživatele obstarat si vlastní data a ten se může dát přímo do analýzy a tvorby vizualizací. Nepřímou cestou lze k aplikaci připojit i vlastní databázový SQL systém (Microsoft SQL Server, PostgreSQL, Oracle) . K tomuto řešení je ovšem třeba využít dalších služeb platformy ArcGIS – ArcGIS Server[3]. Uživatel si může vybrat z velkého množství předpřipravených vizuálních stylů a tím svoji mapu udělat ještě více atraktivnější. Podporuje 4 režimy sdílení vytvořených map – privátní, skupina vybraných uživatelů, všichni uživatelé z vytvořené organizace a veřejné. Pro zobrazení pomocí ArcGIS si uživatel může vybrat z jednoduchého prohlížeče

[1] https://www.esri.com/en-us/arcgis/products/arcgis-online
[2] https://www.arcgis.com
[3] https://enterprise.arcgis.com/en/server/latest/get-started/windows/what-is-arcgis-for-server-.htm

map, prohlížeče scén (obrázek 3.1) zobrazující zeměkouli ve 3D nebo otevření v desktopové ArcGIS aplikaci.

ArcGIS systém podporuje pokročilé začleněni map do vlastních podkladů. První způsob je pomocí kompletní webové komponenty vytvořené v aplikaci ArcGIS Experience Builder[4]. Druhý způsob je vlastní konfigurace pomoci API jazyka JavaScript. Poslední možnost nalezne využití v nativních aplikacích – například v jazyku Java, .NET nebo i při vývoji na iOS a Android, a to pomocí ArcGIS Runtime SDK[5].

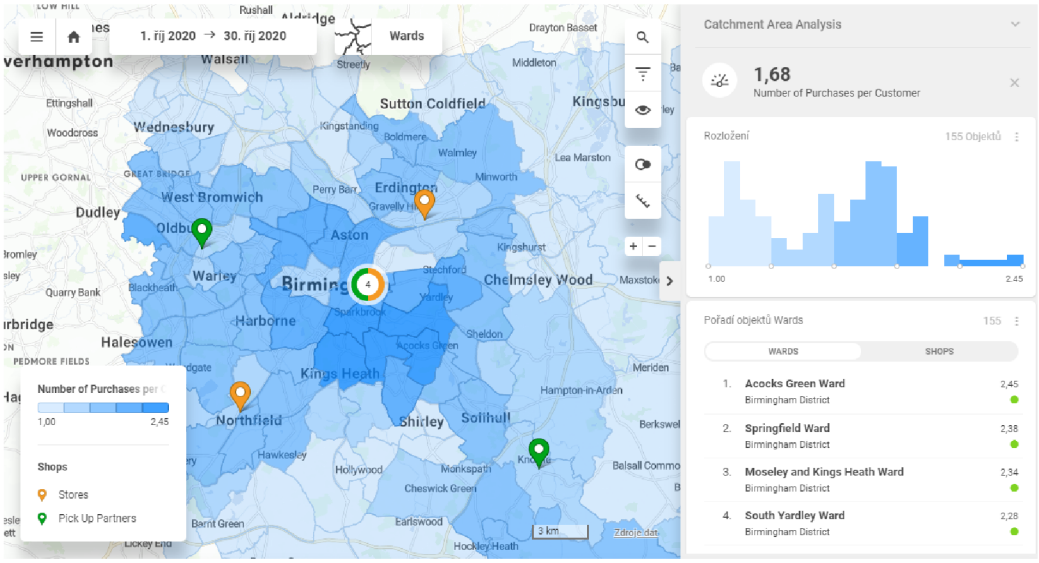


Obrázek 3.1: Náhled varování před počasím v USA, prohlížeč scén ArcGIS.

3.2 CleverMaps[6]

Jedná se o platformu zaměřenou především na analýzu geografických dat zaměřených na obchodování za účelem podpořit obchodní rozhodnutí. Za cíl si dává přiřadit různým úrovním dat kontext zobrazením na mapě. V aktuální verzi neposkytuje grafické webové rozhraní pro konfiguraci, pouze pro zobrazení výsledků. Uživatel si stáhne kontejner založený na technologii Docker[7] ve kterém běží *CleverMaps shell*[8], skrz který si uživatel spravuje svůj prostor. Hlavním prvkem výstupu je pak interaktivní mapa. Kromě zobrazení zpracovávaných dat určitým způsobem do mapy, obsahuje i postranní lištu s analýzou zobrazovaných dat, či možností filtrování a agregování dat. Taktéž disponuje tržištěm s daty, tzv. *Data Marketem*. Zde nalezne uživatel množství aktualizovaných datových sad z velkého množství oblastí. Jmenovitě například údaje o pohybu osob či vozidel, demografické datové sady, datové sady znázorňující kupní sílu apod. Pohled na vizualizovaná data vizualizované pomocí CleverMaps lze vidět v obrázku 3.2. Pro sdílení výsledku může uživatel využít personalizovaný odkaz s přístupovým tokenem, který lze použít ve vlastních webových prezentacích jako zdroj obsahu pro *iframe* element.

[4]https://www.esri.com/en-us/arcgis/products/arcgis-experience-builder/overview
[5]https://developers.arcgis.com/documentation/mapping-apis-and-services/apis-and-sdks/
[6]https://www.clevermaps.io/
[7]https://www.docker.com/
[8]https://clevermaps-internal.atlassian.net/wiki/spaces/CDD/pages/7536890/CleverMaps+Shell

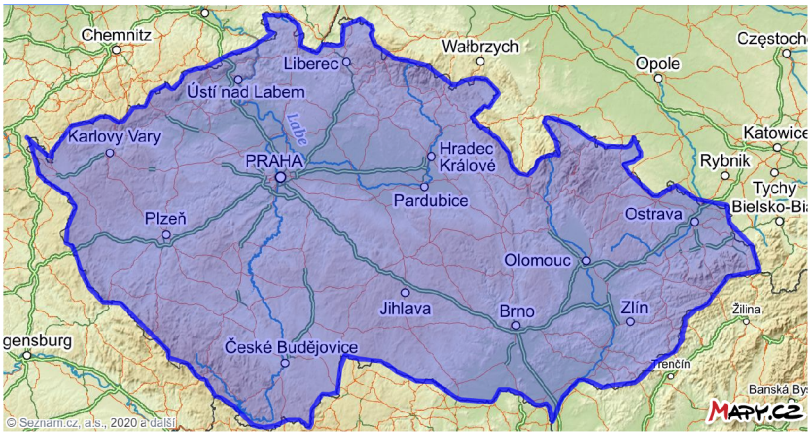


Obrázek 3.2: Vizualizace testovacího projektu znázorňující pomocí kartogramu data z připravené datové sady.

3.3 API Mapy.cz[9]

Společnost Seznam.cz a.s. poskytuje aplikační rozhraní ke svým mapám, které uživateli umožní vložit si element s mapovými podklady na svou stránku. Použití Mapy API je zcela zdarma a je možné jej využít i pro komerční účely. Pro použití je ovšem nutné, aby uživatel ovládal jazyk HTML a JavaScript. Veškerá konfigurace mapového rámce probíhá na straně uživatele pomocí volání metod API, které se na uživatelovu webovou stránku dostanou pomocí takzvaného *loaderu*. Protože se nejedná o informační systém, Mapy API za uživatele nijak neřeší uložení dat, zpracování ani vytvoření prostoru pro následující sdílení. Pro potřeby vizualizace (obrázek 3.3) je zde podpora kreslení vektorových prvků, přidávání vrstev se značkami, vytváření shluků těchto značek nebo načtení a zobrazení souboru formátu GPX[10].

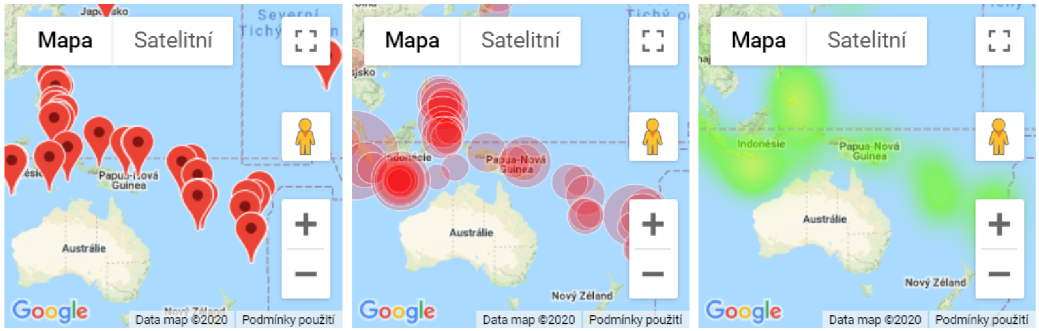
[9] http://api.mapy.cz/
[10] https://www.topografix.com/gpx.asp



Obrázek 3.3: Vyznačení oblasti pomocí vektorového prvku.

3.4 Google Maps JavaScript API[11]

Platforma Google Maps poskytuje sadu jak SDK (*software development kit*[12]), tak API pro práci s mapovými podklady. Našemu zaměření se nejvíce blíží Google Maps JavaScript API, které umožňuje zobrazovat grafické prvky nad vrstvou mapy. Google Maps API se používá téměř totožným způsobem jako API od Seznam.cz – závislost jazyka JavaScript se vloží do zdrojového kódu stránky a pomocí poskytnutých funkcí se upravuje předem určený HTML element. Tímto procesem dojde k začlenění elementu s mapou do vlastní webové prezentace. Ve výčtu podporovaných funkcí se nachází například importování datových vrstev ze souborů ve formátech GeoJSON nebo XML. Takto importovaná data se následně zobrazí jako obrazce, body nebo formou teplotní mapy (viditelné na obrázku 3.4). Opět, stejně jako v případě Seznamu, právě zmiňované API za uživatele nijak neřeší uložení dat, zpracování ani vytvoření prostoru pro následující sdílení. V případě potřeby si ale zkušenější uživatel najde nástroje přímo z ekosystému Googlu sloužící ke zmíněným účelům.



Obrázek 3.4: Grafická vizualizace tří základních typů zvýraznění prvků datové sady pomocí Google Maps JS API.

[11] https://developers.google.com/maps/documentation/maps-static/
[12] SDK je rozsáhlá sada nástrojů pro práci a integraci daného produktu

3.5 ScribbleMaps[13]

Webová aplikace, poskytující v neplacené verzi základní sadu nástrojů. Zde si uživatel může vytvořit vlastní mapu včetně bodů zájmů a různých obrazců. V placené verzi pak přidává import dat ve formátu CSV, GeoJSON nebo KML. Do těchto typů formátů pak zvládá i exportovat. Podle podoby zobrazovaných dat zvládne i export do Shapfile nebo AutoCAD DXF (Drawing Exchange Format). Obsahuje množství funkcí pro analýzu a vizualizaci dat (například vizualizace v obrázku 3.5), jako zobrazování či zvýrazňování dat, agregování dat, přidávání markerů do mapy nebo třeba nalezení konvexní obálky okolo zobrazovaných entit. Aplikace umožňuje sdílení pomocí odkazu, vygenerování kódu pro vložení widgetu s mapou v podobě *iframe* elementu do vlastní webové prezentace, zaslání odkazu ke sdílení emailem nebo uložení aktuálního náhledu do grafického souboru. Čím ScribbleMaps vyčnívá z řady je určitě možnost objednat si fyzickou verzi mapy se všemi grafickými prvky vyobrazenými právě touto aplikací a to až do velikosti plátna 36x24 palců.

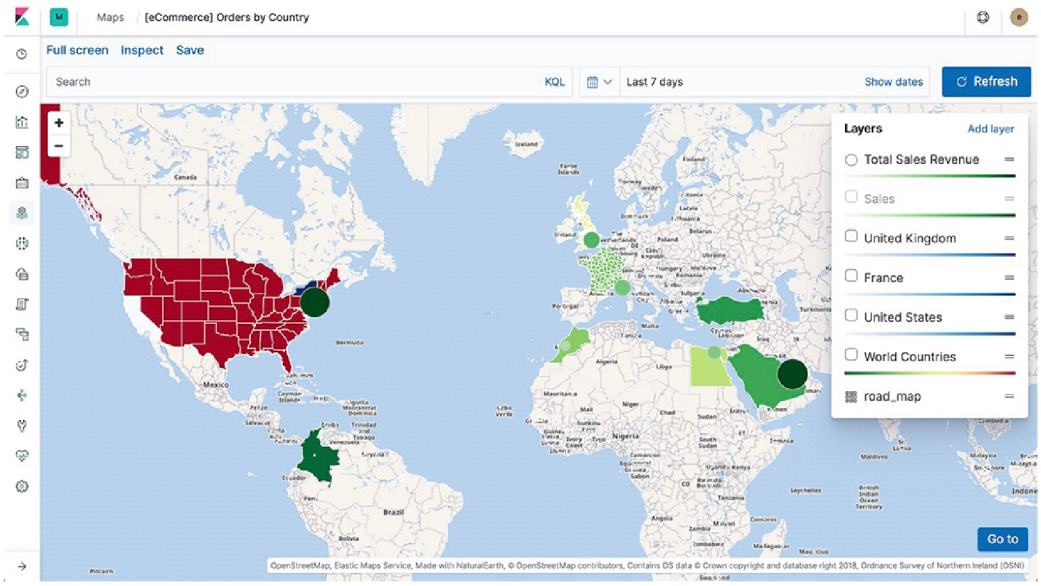
Obrázek 3.5: Zobrazení tématické mapy typu Heatmap ve zkušební verzi aplikace ScribbleMaps.

3.6 Elastic Maps[14]

Elastic maps je nástroj postavený okolo platformy Elastic, nazývanou jako Elastic Stack. Dostupný pomocí aplikace Kibana[15], která slouží jako přístupový bod do oblasti vizualizace dat, jak ve formě desktopové aplikace, tak i v cloudové verzi. Právě Elastic Maps je nástroj zaměřený na zpracovávání a analýzu geografických dat. Jako podklad poslouží geografické vrstvy s 18 úrovněmi detailů – rozdělení na základě světadílů, států, krajů atd. Před vizualizací dokáže vykonávat operace nad daty, jako je filtrování, řazení, dotazování podle textových nebo numerických hodnot a mnohé agregační funkce. Zvládne kombinovat více

[13] https://www.scribblemaps.com
[14] https://www.elastic.co/maps
[15] https://www.elastic.co/what-is/kibana

zdrojů dat najednou a zároveň je i aktualizovat v reálném čase. Elastic Stack může pomoci například i strojovým učením, kdy dokáže hledat anomálie[16] v datových sadách. Taktéž obsahuje velké množství tématických map a grafů. Uživatelům dovolí definovat vlastnosti vizualizace i co se grafické stránky týče (obrázek 3.6). Sdílení vizualizací pak probíhá pomocí funkcionality pro sdílení v nástroji Kibana a to již několikrát zmíněnou metodou – pomocí odkazu nebo HTML elementu *iframe*. Stejně tak jako u ostatních profesionálních řešení, platforma Elastic obsahující nástroj Elastic Maps je placená s možností dohodnutí si zkušební verze.

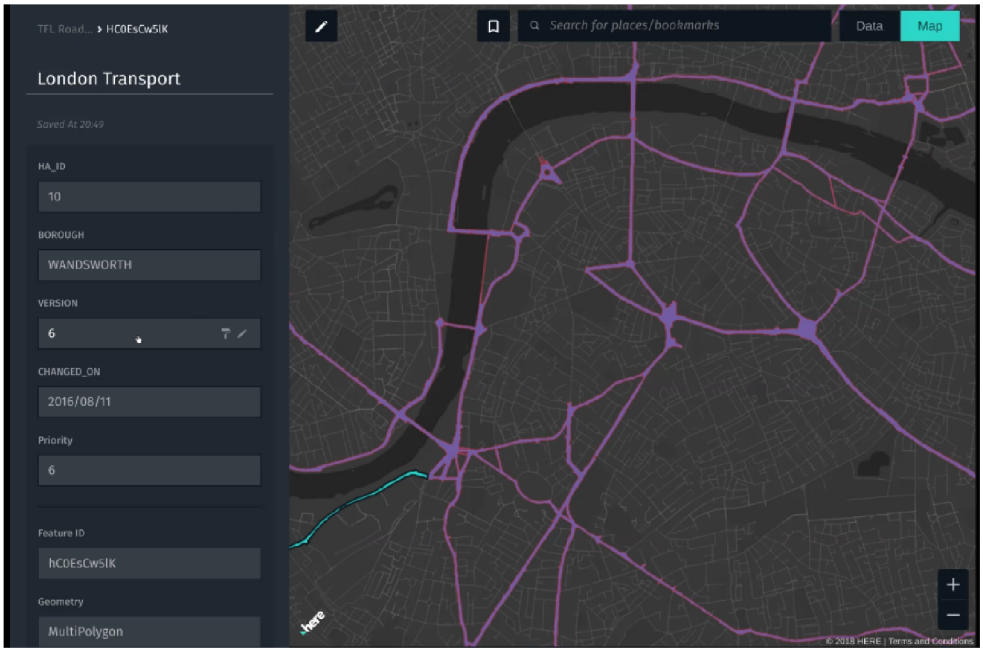


Obrázek 3.6: Pohled na uživatelské prostředí aplikace Elastic Maps zobrazující kombinaci více datových sad.

3.7 HERE Studio[17]

Důležitým produktem z dílny HERE[18] je program HERE Studio – interaktivní webová aplikace, která umožňuje tvorbu map a vizualizací skrze grafické rozhraní bez nutnosti znalosti programování. Nejprve si uživatel zvolí mapový podklad, kterých nalezne v aplikaci na výběr hned několik. Nad tento podklad poté může importovat vrstvy objektů související s geografickými daty z formátů jako GeoJSON, Shapefile apod. Dále je uživateli dovoleno do mapy rovnou i zaznamenávat, a to kreslením vlastních bodů nebo přidáním úseček. Platforma HERE poskytuje funkci HERE Data Layers, tedy množství předpřipravených datových sad ve formátu GeoJSON. Jedná se například o půdorysy budov včetně pojmenování, značení cest, vodních ploch, řek nebo i množství informací o typech půdy. Nad importovanými daty pak lze nastavit podmíněné styly, tedy proces, pomocí kterého se aplikace stará o grafické zobrazení. Takto připravenou mapu (obrázek 3.7) lze i velice snadno sdílet buď pomocí vygenerovaného odkazu, nebo vložením HTML snippetu do vlastních stránek. Vzhledem ke všem funkcím, které tento produkt obsahuje, není překvapením, že je jeho užívání přímo podmíněné zaplacením za licenci.

[16] https://www.elastic.co/what-is/elasticsearch-machine-learning
[17] https://www.here.com/platform/studio
[18] https://www.here.com

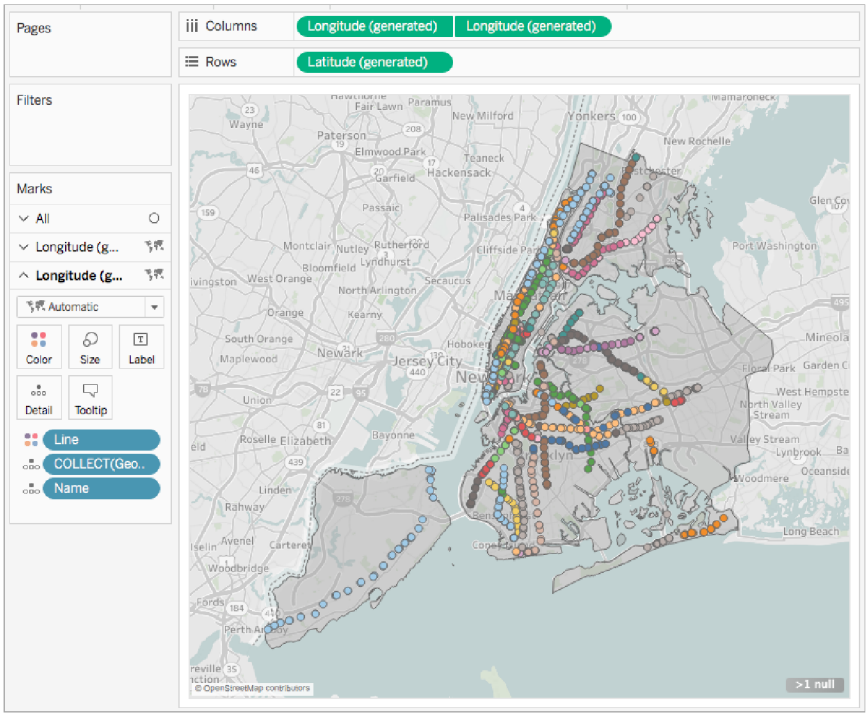


Obrázek 3.7: Grafické uživatelské rozhraní aplikace HERE Studio umožnující editaci a zobrazení geografických entit.

3.8 Tableau[19]

Tableau dokáže jednoduše převést geografická data do interaktivních map, buď automaticky nebo pomocí vlastních geokódů. Obsahuje množství demografických datasetů. Podporuje vizualizaci pomocí množství tématických map, jako jsou kartogramy, mapy značek, teplotní mapy a další. Jako zdroj dat poslouží množství formátů jako jsou Shapefile, KML, GeoJSON nebo TopoJSON. Také podporuje i sloučení více souborů najednou při použití v jedné vizualizaci. Řešení v podobě Tableau podporuje dokonce i připojení vlastní databázi. Na výběr má uživatel z velkého množství jak SQL tak NoSQL databázových systémů. Sdílení výsledků probíhá klasicky, zpřístupněním vizualizace na daném odkazu, který lze buď sdílet nebo dále vkládat. Aplikace Tableau (náhled v obrázku 3.8) je sice placená, umožňuje ovšem zákazníkům otestovat funkcionalitu v bezplatné časově omezené zkušební verzi.

[19] https://www.tableau.com/solutions/maps



Obrázek 3.8: Vizualizace hodnot metodou teček datové sady v prostředí Tableau.[20]

3.9 Shrnutí

Většina z představených řešení si jsou v mnoha ohledech podobná. Obvykle se jedná o placenou aplikaci či službu, umožňující import dat pomocí shodného výčtu podporovaných formátů. Disponuje vlastním úložištěm datových sad, které nabízí uživatelům podle předem stanovených tarifů. Několik nástrojů pak umožňuje i načítání dat v reálném čase, průběžnou aktualizaci datových sad a tak dosáhnutí animování prvků na mapě. Hlavním lákadlem současných řešení je určitě například právě zmíněná interaktivita výsledných vizualizací nebo vizuální zpracování. Grafické prvky na mapě mnohdy doplňují i postranními panely s pohledem na data zasazená do grafů. V čem se ale aspoň pár řešení liší, je postup, jakým je potřeba postupovat pro vizualizaci dat. Ten lze rozdělit buď na práci s příkazovou řádkou a nebo na práci s přehledným webovým uživatelským rozhraním. V poslední části procesu – sdílení výsledných vizualizací, se však funkcionalitou všechna řešení znova setkávají.

Nejrazantnější limitací všech zkoumaných řešení je určitě cena. Stávající cloudová řešení také znamenají to, že místo jednorázové licence se většina ze společností vyvíjejících tato řešení uchýlila k měsíčnímu předplatnému služeb. Pokud byla přítomná bezplatná verze, její funkcionalita zůstala velice omezená a záměr donutit uživatele k nákupu plné verze byl zcela zřejmý. Jako další nedostatek bych viděl možnost importovat datovou sadu z vlastního zdroje. Ať už specifikovaného přes URI nebo klasický databázový systém. Stávající řešení vetšinou spoléhají na interní datové sady nebo pouze na import souborů podporovaných formátů.

[20] https://help.tableau.com/current/pro/desktop/en-us/maps_shapefiles.htm

Kapitola 4

Tvorba informačních systémů

Po teoretické části vizualizace geografických dat je důležité také představit způsoby tvorby informačních systémů, které umožní správu těchto vizualizací. V této kapitole přibližuji, co je to geografický informační systém, z čeho se takový informační systém skládá a jak uvnitř něj probíhá komunikace.

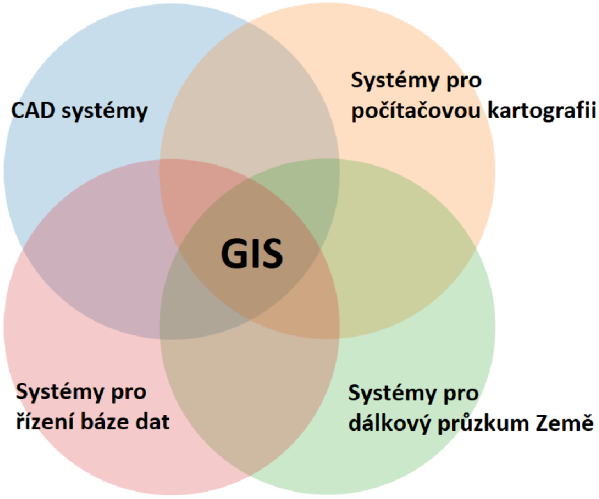
4.1 Geografické informační systémy

Geografický informační systém, neboli zkráceně GIS, je počítačový systém pro správu, ukládání a analýzu prostorových dat – anglicky *geospatial data*. Tyto data musí uchovávat jak údaje o poloze objektu, tak i jeho prostorové rysy. Komponenty, z kterých se GIS skládá, zahrnují hardware, software, data a organizace. Díky rozšíření osobních počítačů a grafických uživatelských rozhraní zaznamenaly GIS zásadní rozkvět již v 80. letech [8]. Mezi jeden z prvních geografických informačních systémů patří kanadský geografický informační systém (*CGIS – Canada Geographic Information System*) obsahující využití krajiny, informace o zemědělství, lesnictví a využívání přírodních zdrojů [11].

4.1.1 Definice GIS

Pro stanovení definice je důležité uvědomit si spojení mezi geografickými informačními systémy a CAD (*Computer aided design*) systémy spolu s počítačovou kartografií, systémy pro řízení báze dat a systémy pro dálkový průzkum Země (znázorněno na obrázku 4.1). Někdy se tvrdí, že geografické informační systémy jsou podmnožinou nebo nadmnožinou těchto systémů [22].

- GIS je jakýkoliv manuálně nebo počítačově založený soubor postupů užívaných k ukládání a manipulování geograficky vztažených dat. [4] – **S. Aronoff**.
- GIS je účinnou kolekcí nástrojů pro sběr, uchovávání, získávání a v případě potřeby transformování a zobrazování prostorových dat reálného světa [6] – **P. A. Burrough**.
- GIS je speciálním případem informačního systému, kde se databáze skládá z popisů prostorově rozložených charakteristik, aktivit a jevů, které jsou v prostoru definovatelné jako body, linie či plochy. Geografický informační systém zpracovává data o těchto bodech, liniích a plochách a to tak, aby je bylo možné využít k odpovědím na dotazy a k analýzám jednotlivých úloh [9] – **K. J. Dueker**.



Obrázek 4.1: Diagram znázorňující vazbu mezi jednotlivými informačními systémy s vlivem na GIS.

4.1.2 GIS a databázové systémy

Pro uchovávání neprostorových dat se obvykle využívá relačních databází. Použití toho zmíněného typu databází pro uložení prostorových složek geografickým dat ale vede k určitým komplikacím. Jedná se například o to, že geografická informace je komplexní, a proto je aktualizace geografických dat prováděna složitými transakcemi, které se týkají většího počtu záznamů v několika tabulkách. Další z problému je, že prostorová data mají proměnnou délku záznamů v závislosti na počtu souřadnic [30]. Pro uložení geografických dat jsou požadovány složitější datové typy, chybějící v klasických relačních databázích. Z tohoto důvodu se jako skvělý kandidát pro uchovávání dat geografických informačních systémů jeví objektové NoSQL databáze.

4.2 Architektury informačních systémů

Od vzniku prvních informačních systémů již uplynulo mnoho let a tak se již ustálily určité principy, normy a doporučení. Tyto principy je vhodné dodržovat například při návrhu architektury či přiřazování zodpovědnosti určitým částem systému.

4.2.1 Klient-server architektura

Architektura klient-server je jeden z typů architektur informačních systémů. Cílí jak na zvýšení výkonu systému jako celku, tak na snížení zátěže na počítač uživatele. Oproti staršímu přístupu, kdy se na straně klienta prováděly všechny výpočty a serverová část sloužila pouze pro potřeby poskytování dat, se nyní na uživatelově straně vyskytuje pouze tenký klient. Data se pak zpracovávají tam, kde jsou uložena a toto místo se nazývá server [5].

4.2.2 Třívrstvá architektura

Třívrstvá architektura vychází z architektury klient-server, ale přináší mnohá vylepšení. Názvy jednotlivých vrstev jsou: klientská, aplikační a datová. Množství aplikační logiky,

která byla doposud prováděna na straně uživatele v klientu, se přesunulo do aplikační vrstvy na straně serveru. Aplikační vrstva tak komunikuje s datovou, provede požadované výpočty a předává klientské vrstvě potřebná koncová data, například ve formátu JSON nebo XML. Zavedením této architektury se také ulevilo i síťovému provozu.

4.2.3 Serverová část

Serverová část, neboli *backend* či *server tier*, je jádro každého informačního systému. Z popisu třívrstvé architektury již víme, že backend provádí výpočty, komunikuje s databází a obsluhuje požadavky z klientů. Nejčastějšími programovacími jazyky při vývoji serverové části jsou Java, Ruby, PHP, JavaScript a Python, přičemž každý ze zde jmenovaných obsahuje řadu knihoven pro tvorbu webových aplikací, včetně nástrojů pro přístup k databázovým systémům.

4.2.4 Klientská část

Při tvorbě webových aplikací dnes jasně dominuje použití jazyka JavaScript, ať už se jedná jen o podpůrné skripty, které mají za úkol udělat statické stránky dynamickými, nebo o kompletní řešení frontend části pomocí jednoho z populárních frameworků jazyka JavaScript jako jsou React[1], Angular[2] nebo Vue.js[3]. Vue.js je vyvíjen výhradně open-source komunitou, zatímco na Angularu mají významný podíl zaměstnanci Googlu a na Reactu zaměstnanci Facebooku. Angular používá TypeScript a dělí logiku HTML a TypeScriptu od sebe, pro React se používá charakteristický JavaScript nazývaný JSX, který kombinuje logiku HTML a JavaScriptu, Vue používá běžný JavaScript a rozděluje zvlášť část HTML a JavaScriptu.

CSS Frameworky

Cascading Style Sheets (CSS) je účinný způsob jak ovlivnit vizuální stránku HTML dokumentu. Z důvodu ulehčení a částečně i urychlení vývoje webových aplikací začaly vznikat CSS frameworky. CSS framework je specializovaná knihovna s množstvím předdefinovaných stylů usnadňující práci například díky pomocným třídám pro rychlé pozicování elementů, typografii, barevnými schématy, ikonami či množstvím přichystaných elementů v jednotném vizuálním stylu, jako jsou tlačítka, formuláře a jiné interaktivní prvky.

4.2.5 Komunikace uvnitř systému

Pro komunikaci klientské a serverové části je potřeba mít domluvené rozhraní, aby si obě části systému rozuměly. Určitým standardem v této oblasti se stalo **REST** (*representational state transfer*) rozhraní založené na požadavcích (*request*) a odpovědích (*response*). Server má předdefinované tzv. *endpoints*[4] na kterých očekává HTTP požadavek typu GET/POST/PUT/DELETE [17]. Každý z typů požadavků má pevně definovanou operaci, kterou serverová část vykoná, a následně vrátí odpověď. Modernější přístup v podobě **GraphQL**[5] přináší vylepšení, kdy se v HTTP požadavku definuje i požadovaná forma dat v odpovědi. GraphQL je dotazovací jazyk určený k užití klientskými aplikacemi poskytnutím intuitivní

[1] https://reactjs.org
[2] https://angular.io
[3] https://vuejs.org
[4] Adresa URI používaná k přístupu ke zdrojům či jejich editaci
[5] https://graphql.org

a zároveň flexibilní syntaxe a systému pro popis požadovaných dat a interakcí. Klient se tak dotazuje pouze na data, která využije a uleví tak síťovému provozu, ale také sobě, protože se nemusí zabývat daty, která momentálně nepotřebuje.

4.2.6 Vizualizace dat

Nejjednodušší možností, jak graficky vizualizovat data na webu, a to nesouvisí pouze s geografickými daty, je přímo využít způsobu, jakým jsou tvořeny HTML dokumenty. Tím je myšleno, vytvořit klasický HTML dokument, který poslouží jako šablona a kontejner pro data. Pomocí kaskádových stylů ho přetvořit tak, aby odpovídal požadavkům na výslednou vizualizaci. Pokud je potřeba dosáhnout i interaktivity a vizuálních efektů, kromě pokročilých metod poskytnutých kaskádovými styly je možné využít i jazyka JavaScript, případně knihovny pro zacházení s objekty DOM (*Document Object Model*) jQuery[6]. Další možností je využít HTML element *canvas* (plátno) do kterého lze pomocí volání určitých metod jazyka JavaScript kreslit základní geometrické tvary i text. Příklad vykreslení jednoduché kružnice do elementu plátna lze vidět ve výpisu 4.1.

```
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.beginPath();
    ctx.arc(95, 50, 40, 0, 2 * Math.PI);
    ctx.stroke();
</script>
```

Výpis 4.1: Příklad vykreslení kružnice do HTML elementu *canvas*.

Pokročilejší přístup je použití formátu SVG (*Scalable Vector Graphics*) uvnitř stejnojmenného HTML elementu a grafiku tvořit přímo v něm. Jedná se o plnohodnotný vektorový formát s podporou geometrických tvarů, křivek, pokročilou prací s textem, průhlednosti a podobně. Grafické objekty a s nimi související informace, jsou pak uloženy v XML formátu odpovídajícímu oficiálně vydané specifikaci [3].

Doposud popsané způsoby se ovšem stále drží na té nejnižší úrovni, které lze ke tvorbě grafiky ve webovém prostředí využít. Pro zefektivnění práce tak vzniklo množství knihoven, založených na zmíněných přístupech. Jedná se například o knihovnu D3.js[7] – knihovnu jazyka JavaScript, která je založená na práci s dokumenty zaměřenými na data. Tyto dokumenty pak dokáže převést na grafické objekty použitím HTML, CSS a SVG. V základu obsahuje rozmanitou škálu funkci pro tvorbu grafiky a grafů, včetně množství nádstaveb zaměřených na konkrétní účel. Z pohledu vizualizace geografických dat se však stále jedná o nízkoúrovňové řešení.

Geografické knihovny

Nabízí se tak využít některou z knihoven, které jsou přímo určené k tomuto úkolu. Jedná se třeba o knihovnu Leaflet[8], jednu z předních knihoven jazyka JavaScript pro tvorbu interaktivních map, vhodnou i pro použití na mobilních zařízeních. Tato knihovna obsahuje množství funkcí pro práci s daty nad mapovými podklady. Hlavní přidanou hodnotou je

[6] https://jquery.com/
[7] https://d3js.org
[8] https://leafletjs.com

rozhraní, kterým vývojáře odstíní od nízkoúrovňového zacházení s grafickými objekty. Například místo kreslení na plátno pomocí souřadnic pixelů tak umožní definovat pozici pomocí klasického geografického souřadnicového referenčního systému. Jak vypadá v praxi vykreslení kružnice do elementu mapy, lze vidět ve výpisu 4.2. Alternativou ke knihovně Leaflet pak může být brána knihovna Mapbox GL JS[9]. Nejzásadnějším rozdílem je využití technologie WebGL[10] pro vykreslování interaktivních map. Stejně jako v případě Leaflet obsahuje vysokoúrovňové rozhraní pro práci s geografickými daty.

```
var circle = L.circle([51.508, -0.11], {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5,
    radius: 500
}).addTo(mymap);
```

Výpis 4.2: Příklad vykreslení kružnice na určitých souřadnicích pomocí rozhraní knihovny Leaflet.

4.3 Databázové systémy

Pro efektivní využívání informačního systému je třeba mít uložené velké množství dat na určitém místě tak, aby bylo možno tyto data kdykoliv získat a následně je zpracovávat. Tomuto místu se říká databáze. K uložení dat do databáze existují různé postupy a právě podle způsobu, jakým jsou data organizována, lze databázové systémy dělit, například na relační a nerelační databázové systémy [13]. S takto uloženými daty poté nakládají systémy řízení báze dat (SŘBD, anglicky *database management system* DBMS), jako jsou například PostgreSQL, MySQL nebo MongoDB.

Pro potřeby komunikace backend části s konkrétním databázovým systémem lze použít ovladače (*drivers*) nebo konektory napsané v daném programovacím jazyku. Tyto konektory tvoří mezivrstvu v komunikaci backend části a databázového systému poskytující většinou přehledné, snadno použitelné a zdokumentované API pro správu dat v databázi. Při použití frameworků s podporou ORM[11] může dojít i k úplnému odstínění potřeby znalosti jazyka SQL, kdy se ORM automaticky stará o mapování dat z tabulek a konverzi datových typů na objekty programovacího jazyka.

4.3.1 Relační databáze

Relační databáze jsou založeny na relačním modelu, intuitivním, přímočarém způsobu vyjádření dat v tabulkách. V relační databázi je každý řádek v tabulce záznam s jedinečným ID nazvaným klíč. Sloupce tabulky obsahují atributy dat a každý záznam má obvykle hodnotu pro každý atribut, což usnadňuje navázání vztahů mezi datovými body [27]. Mezi zástupce relačních SŘBD patří PostgreSQL[12], Oracle Database[13] nebo MySQL[14] a MariaDB[15].

[9] https://docs.mapbox.com/mapbox-gl-js/api
[10] https://www.khronos.org/webgl
[11] Objektově relační mapování
[12] https://www.postgresql.org
[13] https://www.oracle.com/database/technologies
[14] https://www.mysql.com
[15] https://mariadb.org

4.3.2 Nerelační databáze

Nerelační databáze je databáze, která nepoužívá tabulkové schéma uložení dat, které se nachází ve většině tradičních databázových systémů. Místo toho nerelační databáze používají model úložiště, které je optimalizováno pro konkrétní požadavky typu uložených dat. Data mohou být například uložená jako jednoduché páry klíč/hodnota, jako dokumenty JSON nebo jako graf skládající se z okrajů a vrcholů [32].

Pojem NoSQL odkazuje na úložiště dat, která nepoužívají SQL pro dotazy nad daty, místo nichž se používají příkazy zadané v programovacím jazyku. V praxi NoSQL znamená nerelační databáze, i když mnoho z těchto databází podporuje dotazy kompatibilní s SQL. Nicméně podkladová strategie provádění dotazů je obvykle velmi odlišná od způsobu, jakým tradiční relační SŘBD spustí stejný dotaz SQL [32]. Mezi zástupce nerelačních SŘBD patří třeba dokumentově zaměřený MongoDB[16] nebo grafový Neo4J[17].

4.3.3 Databáze časových řad

Databáze časových řad (anglicky *time series database*, TSDB) je databáze optimalizovaná pro časově označená data nebo data časových řad. Data časových řad jsou měření nebo události, které jsou sledovány, monitorovány a agregovány v průběhu času. Mohou to být metriky serveru, monitorování výkonu aplikací, síťová data, data senzorů, události, počty kliknutí nebo mnoho dalších typů analytických dat. Mezi vlastnosti, díky nimž se data časových řad velmi liší od ostatních datových úloh, patří správa životního cyklu dat, sumarizace a skenování velkého množství záznamů [18]. Jako příklad databáze časových řad můžeme použít InfluxDB[18] nebo TimescaleDB[19].

4.3.4 Metadata

Metadata jsou informace o tom, co popisovaná data obsahují a kde se nacházejí. V případě relačních databází metadata odkazují na informace o jejich schématu a všechny ostatní informace týkající se přístupu, úložiště, integrovaných funkcí nebo jakékoli další informace o prvcích databáze nebo použití. Schéma databáze označuje seznam sloupců tabulek, včetně jejich datových typů. Právě získat toto schéma bude zásadní pro navrhovaný informační systém v případě použití externího databázového systému jako zdroje datových sad. Následně již nebude problém sestavit odpovídající SQL dotaz vracející data ve formě přijatelné pro zpracování.

Při práci s NoSQL databázemi, kdy se podoba objektů může dynamicky měnit, není přítomno pevně dané schéma. V takovém případě, jako například při použití MongoDB, nezbývá nic jiného než si objekty manuálně osahat, a zjistit tím všechny klíče každého z objektů.

[16] https://www.mongodb.com
[17] https://neo4j.com
[18] https://www.influxdata.com
[19] https://www.timescale.com

Kapitola 5

Analýza

V této části se zaměřím na řešení problému geografických informačních systémů hlavně z pohledu teoretické konkurence zmíněné v druhé kapitole diplomové práce. Provedu analýzu předpokládaných person využívajících tento systém včetně jejich požadavků. Dále analýzu existujících nástrojů z pohledu přívětivosti uživatelského rozhraní, použití vlastních dat, tvorby výstupů a jejich typy nebo možnosti sdílení na externím místě, jako je vlastní webová prezentace. Za tímto účelem jsem využil jak volně dostupné uživatelské dokumentace, tak i možnosti práce se systémy pomocí demo náhledů nebo zdarma registrovaných uživatelských účtů.

5.1 Definice cílových uživatelů systému

Vzhledem k tomu, že se zvolená problematika týká vizualizace dat, a to pouze těch s geografickým kontextem, lze očekávat, že potenciálním uživatelem nebude každý. Většinu uživatelů teoretických cílových skupin však spojuje jedna věc, kterou je přístup k geografickým datům, potřeba je vizualizovat a následně i sdílet. Nehraje přitom roli, zda se přímo jedná o vlastníky geografických dat, nebo pouze o uživatele, kterým budou tyto data nasdílena pomocí funkcionality v informačním systému.

Uživatele informačního systému bych rozdělil do následujících třech skupin.

- **Tvůrci** – již zmíněná část lidí, vyznačující se přístupem ke geografickým datům. Jejich úkolem je využít tyto data a vytvořit z nich vizualizace k dalšímu zpracování. Tito uživatelé se orientují v tvorbě tématických map, mají zkušenosti jaký typ mapy použít, či jak data agregovat pro dosažení chtěného výsledku.
- **Analytici** – skupina lidí, kteří mají za úkol tyto data analyzovat a odvodit z nich důsledky. Tento proces samozřejmě počítá s existencí první skupiny. Systém tak může spolehlivě sloužit i lidem, kteří získali hotové tématické mapy a pomocí nové úrovně vizuálního zpracování mohou z dat vyvodit nové závěry.
- **Konzumenti** – tedy uživatelé, kteří již přímo spoléhají na vytvořené vizualizace. Hlavní upotřebení zde vidím například při představování závěrů z vytvořených vizualizací dalším jedincům či široké veřejnosti, doposud nezúčastněných procesu vizualizace geografických dat.

5.2 Požadavky na systém

Požadavky na výsledný geografický informační systém popíši v pořadí, v jakém jsou představeny persony z definice cílových uživatelů. Nejdůležitějším z požadavků je určitě do jisté míry generalizovaný požadavek na vizualizaci geografických dat. Ten bude řešen čistě pomocí knihovny Geovisto. Následně se přidávají požadavky na správu dat. Jedná se o prostor k uchování a správu datových sad, konfigurací a také uživatelských dat, které jsou spojené s účtem uživatele. Co se týká správy geografických dat, nastávají zde požadavky importu rozdílnými způsoby z co největšího počtu zdrojů. Se správou také souvisí možnost kooperace více uživatelů při tvorbě těchto datových sad a konfigurací, nebo znovupoužití vytvořených entit mezi uživateli. Této funkcionality pak využije především skupina analytiků. Ti také mohou mít určité požadavky na orientaci v systému, hledání a řazení ve velkém počtu přístupných entit by tak mělo být samozřejmostí. Skupina analytiků by měla umět systému porozumět i například bez nutnosti teoretické znalosti formátu uchovávaných dat. Proto by měl být systém – hlavně pak uživatelské rozhraní, dostatečně přívětivé. Posledním mnou identifikovaným požadavkem je pak možnost sdílení vytvořené práce a exportu dat.

5.3 Příklad užití

Typickým příkladem použití tohoto informačního systému by byl okamžik, kdy uživatel potřebuje zvýšit úroveň porozumění z jeho dat. V takovém případě může využít tento informační systém. V procesu vizualizace si nejdříve vytvoří datovou sadu. Toho docílí následujícími způsoby:

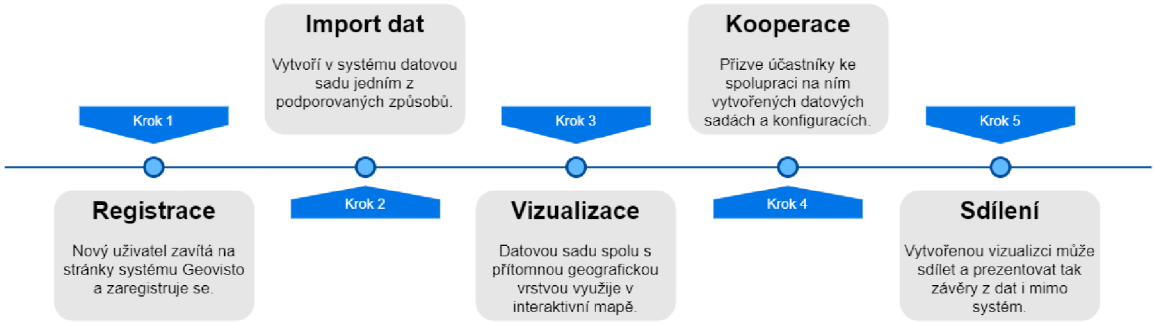
- vloží data formou prostého textu,
- nahraje soubor v jednom z podporovaných formátů,
- připojí existující databázi jako zdroj dat.

V případě potřeby vytvoří vlastní geografickou vrstvu nebo využije jednu z volně dostupných. S takto připravenou datovou sadou může vytvořit konfiguraci, ve které vybere zdroj dat, geografickou vrstvu a následně zvolí vhodný způsob vizualizace.

Další využití nalezne systém v případě, kdy je potřeba sdílení vytvořených vizualizací. Například pokud doposud uživatel vytvářel vizualizace přímo pomocí aplikace Geovisto ve svém soukromém lokálním prostředí a rozhodl by se tyto vizualizace sdílet. Nejjednodušší způsob jak toho dosáhne, bude postupovat podle dřívějšího popisu použití s prováděním kroků, které si již vyzkoušel lokálně. Proběhne tak duplikace jeho lokálního projektu do informačního systému. Následně, při zobrazení detailu konfigurace, uvidí tlačítko sloužící pro vygenerování permanentního odkazu pro sdílení. Pod tímto odkazem bude jeho vizualizace přístupná a zároveň ji tak může vkládat do různých míst na webu pomocí HTML elementu *iframe*. Očekávané pořadí událostí, v jakém bude uživatel se systémem zacházet, lze vidět na obrázku 5.1.

5.4 Existující řešení

Dá se očekávat, že nápad na vizualizaci geografických dat není unikátní nová myšlenka, tedy, že se nejedná o jediný systém svého druhu. Souhrn existujících řešení jsem již popsal ve třetí kapitole, proto se nyní zaměřím na analýzu těchto řešení ve třech kategoriích – uživatelské rozhraní, import dat a sdílení výsledku.



Obrázek 5.1: Předpokládaný průběh zacházení se systémem Geovisto.

5.4.1 Uživatelské rozhraní

Hlavním prvkem bývá část s vyobrazenými mapovými podklady a vrstvou vizualizovaných dat. Poté je obvykle obrazovka doplněna hlavní nabídkou v podobě postranního panelu, který se zobrazí v plné velikosti až poté, co nad ním uživatel podrží kurzor myši, do té doby se zobrazuje pouze v podobě úzkého proužku s piktogramy podsekcí. Tímto způsobem, kdy je postranní panel schovaný po většinu času a nezabírá tím místo na obrazovce, může získat uživatel širší pohled na data.

Správa systému a dat probíhá typicky buď pomocí podstránek nebo modálních oken. Výhodou modálních oken oproti samostatným podstránkám je neustálý přehled o tom, kde se uživatel nachází. Zobrazí-li se mu dialogové okno nad pracovní plochou, uživatel nenabývá dojmu, že se posunul v pomyslném stromu informačního systému na novou úroveň. Pro jednoduchost použití je také vhodné odstínit uživatele od přímé práce s databází, například zaměnit tvorbu SQL dotazů za interaktivní formulář pro výběr sloupců tabulek, přičemž rozhraní reflektuje seznam zvolených položek a na základě vybraných dat se mění v reálném čase.

Rozdílný příklad rozhraní lze najít například u produktu CleverMaps, kde konfigurace projektu probíhá čistě skrze příkazovou řádku. Použití takového způsobu správy dat není moc uživatelsky přívětivé a vyžaduje aspoň základní znalosti pro práci s příkazovou řádkou systému Linux.

5.4.2 Import dat

Při pohledu na zmíněné geografické informační systémy se určitým standardem stalo sdílení a v tomto případě i import dat pomocí formátu GeoJSON, případně import řádků a sloupců v podobě jednoduchých tabulkových dat. Například systém CleverMaps je zcela závislý na formátu JSON, který je použit i pro reprezentaci pomocných dat. Další systémy jako jsou ScribbleMaps nebo ArcGIS Online, umožňují uživateli používat i data uložená ve formátu KML. Zde ale oproti formátu JSON, který přenechává názvy atributů a typy objektů zcela v uživatelské režii, nastává problém manuálně vytvořených značek, které nejsou definovány ve schématu KML. Je tak potřeba používat pouze standardizované KML elementy a případná rozšíření formátu musí být řešena alternativní cestou. Také systém Elastic Maps umožňuje uživateli import vlastních dat ve formátu GeoJSON, zároveň ale nabádá k použití nástroje GDAL[1], který v případě potřeby dokáže transformovat geoprostorová data mezi více než stovkou rozdílných formátů.

[1] https://gdal.org

5.4.3 Sdílení vizualizací

Vytvořené vizualizace jdou sdílet různými způsoby. Prvním je kolaborace účastníků na projektu, kdy vlastník přizve ostatní uživatele ke spolupráci a ti tak mohou zobrazovat nebo editovat projekt. Předpokladem takového řešení ovšem bývá podmínka registrace do systému, což na uživatele požadující pouze náhled vytváří určitý tlak.

Druhým způsobem je zpřístupnění vizualizace na určité URL adrese kterou lze sdílet. Po zobrazení takovéto adresy se sdílená vizualizace zobrazí v celé velikosti okna a díky tomu lze touto cestou zobrazený obsah jednoduše přidat do vlastních webových stránek. Takto sdílené vizualizace ale postrádají možnost editace, jelikož by vlastník neměl žádný způsob jakým zjistit, kdo přesně provedl možné změny.

Třetím, ale poněkud pokročilejším způsobem co se správy týče, je možnost uvolnit balíček nástrojů API, který si vývojář zakomponuje do svého webu. Samozřejmě potřebný kód odpovídá jisté šabloně, takže náročnost na režii lze do jisté míry omezit. V nejlepším případě až na pouhé zkopírování kódu a vložení na stránku. Hlavní výhodou je zde úleva v komunikaci mezi servery daného systému, kdy z řetězce požadavků odpadne zátěž na frontendovou část, protože vykreslovací jádro bude přímo zpracováváno na serveru uživatele. Zůstává tak pouze potřeba komunikace s backendovou částí pro získání dat.

Posledním způsobem sdílení je export výsledků do dokumentu PDF, podporováno například v ArcGIS Online, nebo export do jednoho z formátů grafických souborů jako jsou PNG, popřípadě JPEG. V obou těchto případech pak uživatel sdílí pouze kopii vizualizace, přichází zcela o interaktivitu webové prezentace a podle typu formátu, tedy i způsobu uložení dat o určitý stupeň detailů.

Obvyklé řešení problému sdílení vytvořeného projektu použité v praxi, například z produktu HERE Studio, je takové, že vytvořená vizualizace se zpřístupní na vygenerovaném odkazu. Zároveň se zobrazí také *snippet* pro umístění na vlastní webové stránky pomocí *iframe*. Stejným způsobem postupuje i většina ostatních zmíněných produktů.

5.5 Závěr

Dříve zmíněné informační systémy a aplikace poskytují stávajícím zájemcům robustní sadu nástrojů a způsobů, jak vizualizovat jejich geografická data. Jedná se vesměs o dva typy řešení. Zaprvé to jsou řešení požadující informatické znalosti a investici většího časového úseku, kdy si uživatel sám sestaví vizualizaci na míru. Za druhé to jsou profesionální řešení pomocí rozsáhlých aplikací, které sice vyřeší množství konfiguračních problémů a vytvoří vizualizaci za pár kliknutí, za tuto službu si ale samozřejmě nechají zaplatit. V obou případech pak tyto nástroje umožňují import dat z různých formátů souborů i následné zpřístupnění vizualizace veřejnosti.

Vybraná řešení již, ale až na výjimky jako je ArcGIS a Tableau, nedisponují možností připojit jako zdroj dat vlastní databázi. Právě na tuto skutečnost bude nutné se při vývoji informačního systému zaměřit a získat tak konkurenční výhodu. Stejně tak by stálo za zvážení, naučit informační systém zpracovávat data přímo ze zdrojů specifikovaných pomocí URI.

Kapitola 6

Návrh řešení

V této kapitole se zaměřím na bližší popis navrhovaného řešení. Zaměřím se na každou z částí připravovaného informačního systému. Dostanu se také k předpokládanému způsobu užívání systému i návrhu grafického rozhraní. K závěru popíši, jakým způsobem budou ukládána data systému včetně návrhu modelů, s kterými bude informační systém pracovat.

6.1 Diagram případů užití

Se systémem budou schopni manipulovat aktéři tří kategorií – Návštěvník, Registrovaný a Administrátor. *Návštěvník* symbolizuje právě příchozího uživatele. Ten je schopen prozkoumat zobrazovací část systému pomocí vizualizace z připravené demoverze datové sady, případně vložit svá data v textové podobě, která se ale nebudou v systému uchovávat. Dále pak provést registraci nebo přihlášení do systému.

Pokročilejším aktérem je *Registrovaný*, tedy již registrovaný a přihlášený uživatel. Dokáže spravovat své datové sady, konfigurace, vrstvy map a uživatelský profil. Správa v určitých případech zahrnuje nahrání souborů na server nebo připojení databáze. Všechny jeho vytvořené elementy může dále sdílet s ostatními uživateli, kteří si nastaví svůj profil na veřejný. Samozřejmostí je pak i vizualizace dat.

V hierarchii uživatelů je nejvyšším aktérem *Administrátor*. Jeho práva zahrnují vše z dříve zmíněných aktérů. Navíc ale dokáže sám spravovat seznam uživatelů, jako například pozastavit akce v systému ze strany jejich účtů. Dále pak sledovat statistiky nebo kontrolovat automaticky generované záznamy o běhu a činnostech systému. Diagram případů užití zahrnující tyto aktéry lze vidět na obrázku 6.1.

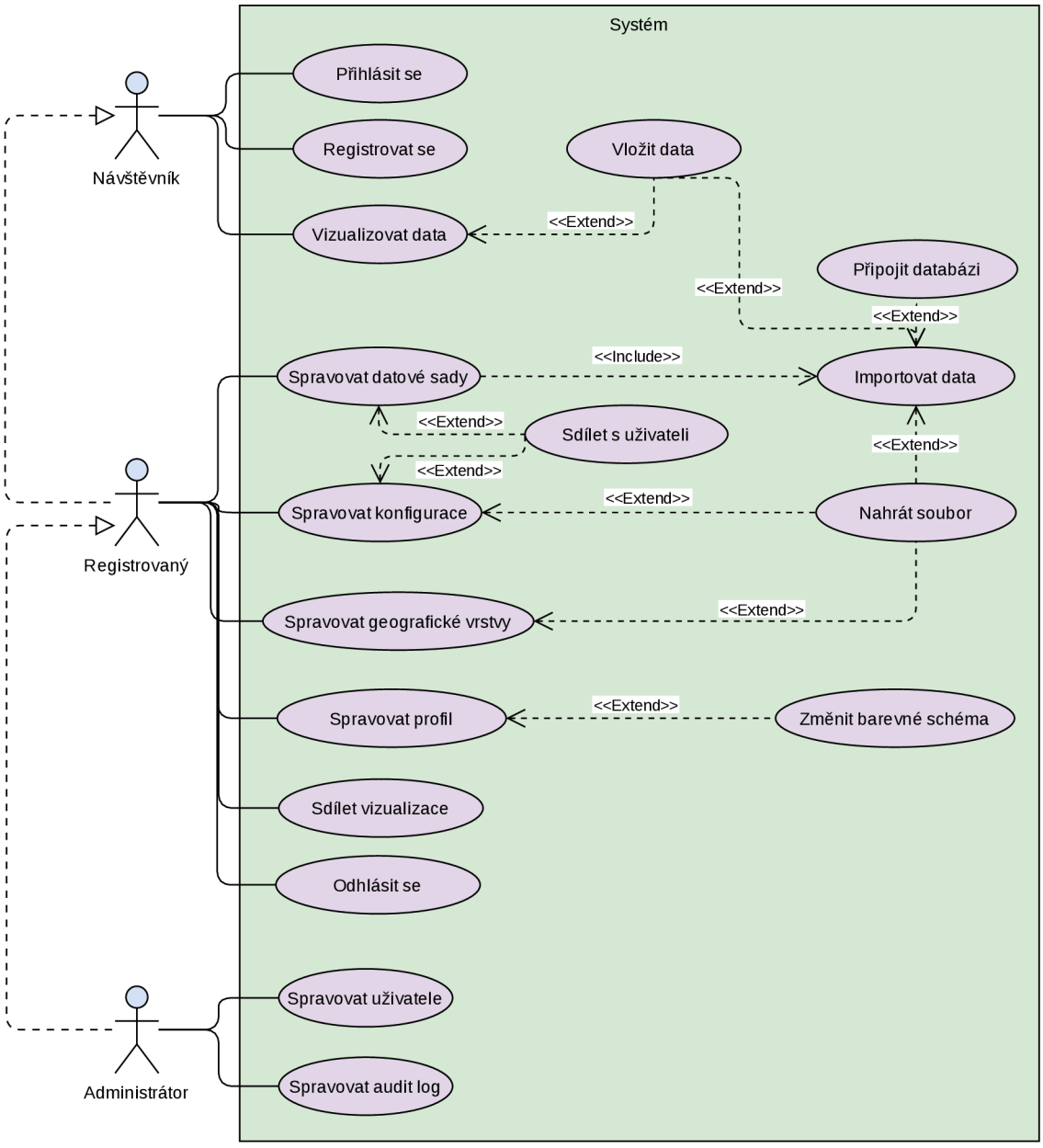
6.2 Serverová část

Serverová část bude styčný bod všech zapojených částí systému. Bude obsluhovat požadavky přicházející z klientské části a také komunikovat s databázovou částí informačního systému i uživatelskými databázemi.

6.2.1 Interakce s klientem

Komunikace s klientskou částí bude probíhat pomocí poskytnuté sady endpointů tvořící aplikační rozhraní. Zamýšlené rozhraní bude založeno na architektuře REST[1], tedy že každý

[1] Representational state transfer



Obrázek 6.1: Návrh diagramu případů užití zachycující hlavní aktéry a jejich akce se systémem.

ze zdrojů má vlastní identifikátor URI[2] přičemž REST definuje čtyři základní metody – *GET, PUT, POST, DELETE* pro přístup k nim [23]. Názvy zdrojů pak budou odpovídat jednotlivým entitám vyskytujících se v modelu systému.

- **/user**
  - GET – Získání dat uživatele
  - POST – Aktualizace dat uživatele
  - **/register**
    - POST – Registrace uživatele
  - **/login**
    - POST – Přihlášení uživatele
  - **/logout**
    - GET – Odhlášení uživatele
  - **/public**
    - GET – Získání kolekce veřejných uživatlů
- **/datasets**
  - GET – Získání kolekce viditelných datových sad
  - POST – Vytvoření nové datové sady
  - **/:id**
    - GET – Získání konkrétní datové sady
    - PUT – Aktualizace konkrétní datové sady
    - DELETE – Odstranění konkrétní datové sady
- **/geolayers**
  - GET – Získání kolekce viditelných geografických vrstev
  - POST – Vytvoření nové geografické vrstvy
  - **/:id**
    - GET – Získání konkrétní geografické vrstvy
    - PUT – Aktualizace konkrétní geografické vrstvy
    - DELETE – Odstranění konkrétní geografické vrstvy
- **/configs**
  - GET – Získání kolekce viditelných konfigurací
  - POST – Vytvoření nové konfigurace
  - **/:id**
    - GET – Získání konkrétní konfigurace
    - PUT – Aktualizace konkrétní konfigurace
    - DELETE – Odstranění konkrétní konfigurace
- **/embeds**
  - **/:id**
    - GET – Získání dat konkrétní vizualizace
    - PUT – Vytvoření unikátního odkazu pro sdílení

[2] Uniform Resource Identifier

Určité endpointy budou pro interakci požadovat jistou míru oprávnění. Serverová část proto bude obsahovat tzv. *middleware* vrstvy, kdy se jedna z nich bude starat o autentizaci požadavků z důvodu ověření oprávnění uživatele. Pomocí této mechaniky pak lze snadno rozlišit, zda daný požadavek odeslal neregistrovaný uživatel, konkrétní registrovaný uživatel nebo dokonce registrovaný uživatel s právy administrátora. Podobná *middleware* vrstva se v systému bude nacházet i z důvodu tvorby záznamu o běhu a činnostech systému. Tento záznam bude prováděn do klasického textového souboru, který bude následně zpřístupněn na zabezpečeném endpointu aplikačního rozhraní pro potřeby analýzy.

6.2.2 Interakce s databází

Komunikace s databázovou částí včetně uživatelských databází, bude probíhat pomocí volně dostupných softwarových konektorů napsaných ve zvoleném programovacím jazyce. Tyto konektory slouží pro vykonávání operací nad databázovým systémem jako je vkládání, získávání a editace dat. Přístupové údaje k hlavní databázi si bude systém držet přímo ve svém prostředí. Přístupové údaje k uživatelským databázím pak bude systém uchovávat společně s ostatními entitami datových sad. Před každým načtením dat tak nejprve vyčte přístupové údaje a pomocí nich získá vždy aktuální data.

Z pohledu bezpečnosti by bylo ideální řešení, kdyby se místo páru uživatelské jméno a heslo uchovával v databázi informačního systému například jen token nebo přístupový klíč. Bohužel, některé databázové systémy, například populární MySQL, tento způsob autentizace nepodporují. Přístupové údaje tak mohou být pouze šifrované, ale zde je vždy nutnost tyto údaje před použitím dešifrovat. Bezpečnější řešení implementace by mohlo být například pomocí tzv. *resource* serveru, tedy dedikované a téměř izolované instance uchovávající přístupové údaje a starající se o získání dat z uživatelských databází. V každém případě ale uživatel musí poskytnout přístupové údaje do databáze. Čím může bezpečnost aspoň trochu zvýšit je, nastavit účtu spojenému s danými přístupovými údaji oprávnění pouze pro čtení.

Po úspěšném připojení k nějaké z databází získá serverová část schéma databáze a podle požadavků ze strany klienta sestaví příslušný dotaz. V případě, že z databáze nebude možno schéma jednoduše vyčíst (například u dokumentových databází), pokusí se systém toto schéma odhadnout získáním všech objektů v kolekci a sestavením jediného objektu zastupující schéma. Následně získaná data pak poslouží jako datové sady a bude je možno prezentovat v aplikaci Geovisto.

6.3 Klientská část

Celá klientská část bude tvořena a měla by se chovat podobně jako tzv. *single-page application* (SPA), neboli jednostránková aplikace. Takové aplikace se vyznačují především přívětivým uživatelským rozhraním a téměř okamžitou odezvou. Toho je docíleno pomocí dynamického načítání obsahu stránek pomocí moderních JavaScript frameworků. Jak již bylo odhaleno v popisu serverové části, komunikace bude probíhat pomocí rozhraní REST. Pro potřeby autentizace bude klientská část posílat včetně požadavků i autentizační token, který bude sloužit pro ověření pravosti a oprávnění.

Klientská část, hlavně tedy její podoba, rozhoduje v mnohých případech zda daný informační systém bude mít mezi uživateli úspěch. V současné době se klade důraz na uživatelské rozhraní stále více a více, protože právě ono rozhraní může uživatele od použití aplikace odradit. Rozhraní informačního systému bude rozdělené na 2 části, postranní panel za-

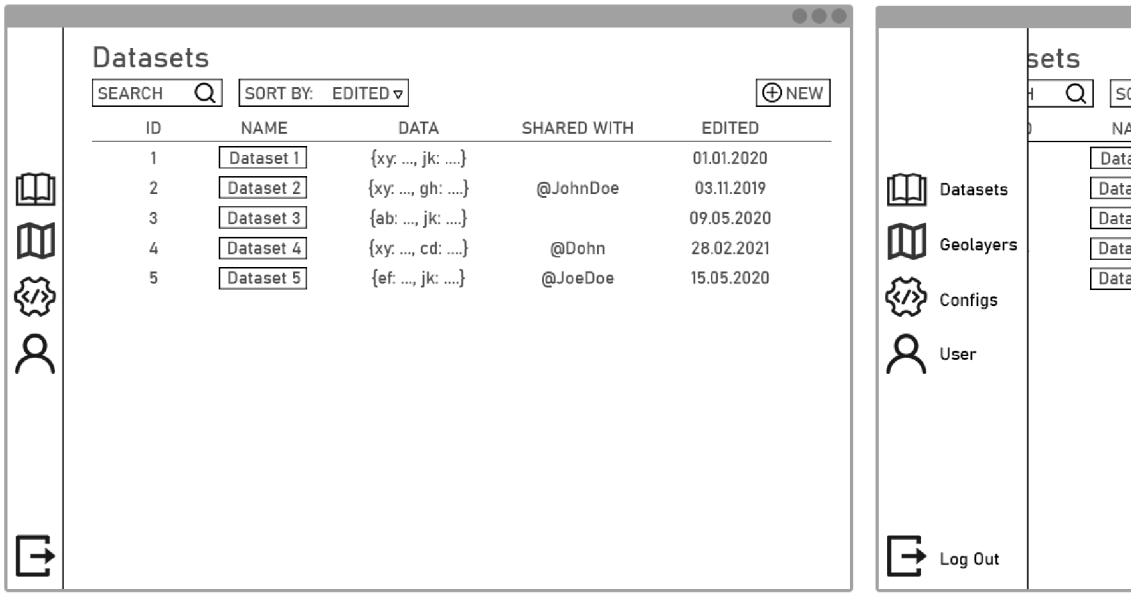
stupující menu a proměnlivá hlavní sekce zobrazující informace podle zvolené podstránky. Při podržení myši nad postranním panelem se panel roztáhne a zobrazí se i textový popis ikon. V případě zobrazení na mobilních zařízeních se postranní panel přesune na dolní část obrazovky.

Celý informační systém se bude skládat z úvodní stránky, jedné administrátorské stránky, stránky pro sdílení a čtyř hlavních sekcí pro každou entitu. Jmenovitě to budou sekce pro správu datových sad, pro správu konfigurací, pro správu vrstev a uživatelská sekce. Úvodní stránka každé sekce pro správu zobrazí uživateli přehled všech odpovídajících objektů, ve kterém se může zanořit do zobrazení detailu vybraného objektu. Přidávání nových objektů bude probíhat v modálním okně na stávající stránce. Nahrávání souborů proběhne buď pomocí funkcionality *drag and drop* (přetažení souboru do vymezené oblasti v okně prohlížeče) a nebo vybráním souboru pomocí dialogového okna.

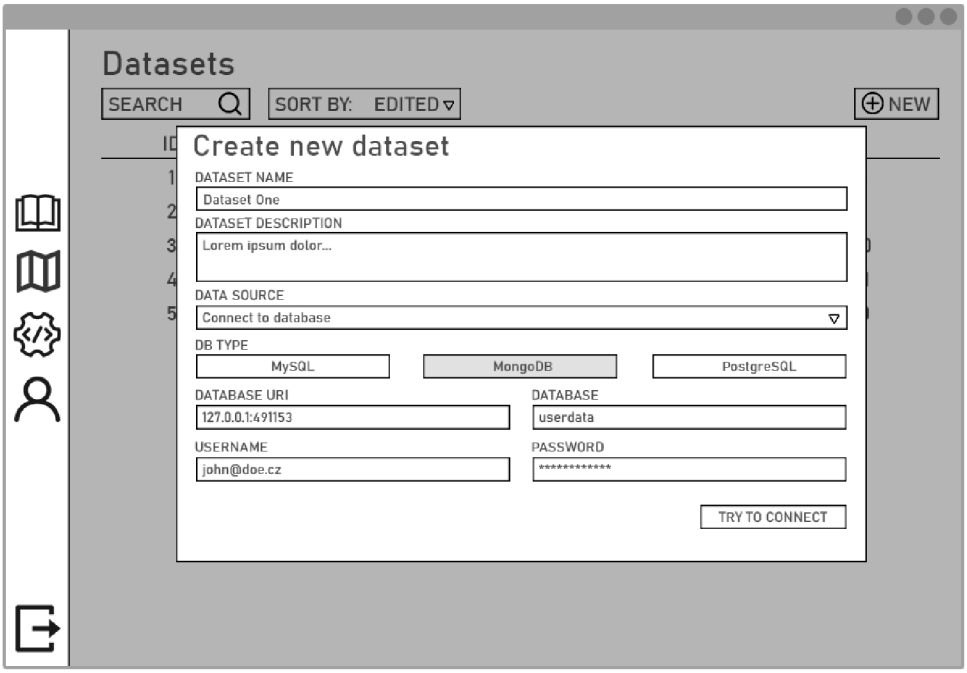
- **Úvodní stránce** bude dominovat sekce s mapou a vizualizace předem připravené datové sady. Konfiguraci vizualizace bude moct návštěvník informačního systému jednoduše měnit, stejně tak i textovou podobu testovací datové sady. Tu bude možné zobrazit nebo změnit v postranní části stránky. V případě vložení vlastní datové sady bude uživatel nucen ručně reflektovat změny i v konfigurační části vizualizace. Na úvodní obrazovce se pak budou nacházet i tlačítka pro registraci a přihlášení uživatele.

- **Administrátorská stránka** bude obsahovat informace o informačním systému například počty entit vyskytujících se v systému, včetně určitých sledovaných vlastností o každém z objektu. Bude tak třeba možné zjistit zda se využívají funkce jako připojení vzdálené databáze, sdílení mezi uživateli nebo zpřístupnění vizualizace pro vložení do vlastních stránek. Poslední funkcionalitou administrátorské stránky bude možnost vzdáleně analyzovat logy serverové části systému.

- **Sekce datových sad – rozpis**, zobrazí rozpis ve formě tabulky (lze vidět v obrázku 6.2), ze které bude možné ihned vyčíst důležitá data o datové sadě. Tabulka bude podporovat vyhledávání i řazení dle hodnot ve vybraných sloupcích. Vytvoření nové datové sady proběhne pomocí modálního okna umožňujícího dva scénáře. První scénář uživatele provede připojením vlastní databáze (obrázek 6.3), výběrem jedné z tabulek a zvolením dimenzí dat. V druhém scénáři mu bude umožněno vložit data ve formě prostého textu či pomocí nahrání souboru z jeho lokálního úložiště do prostředí informačního systému.



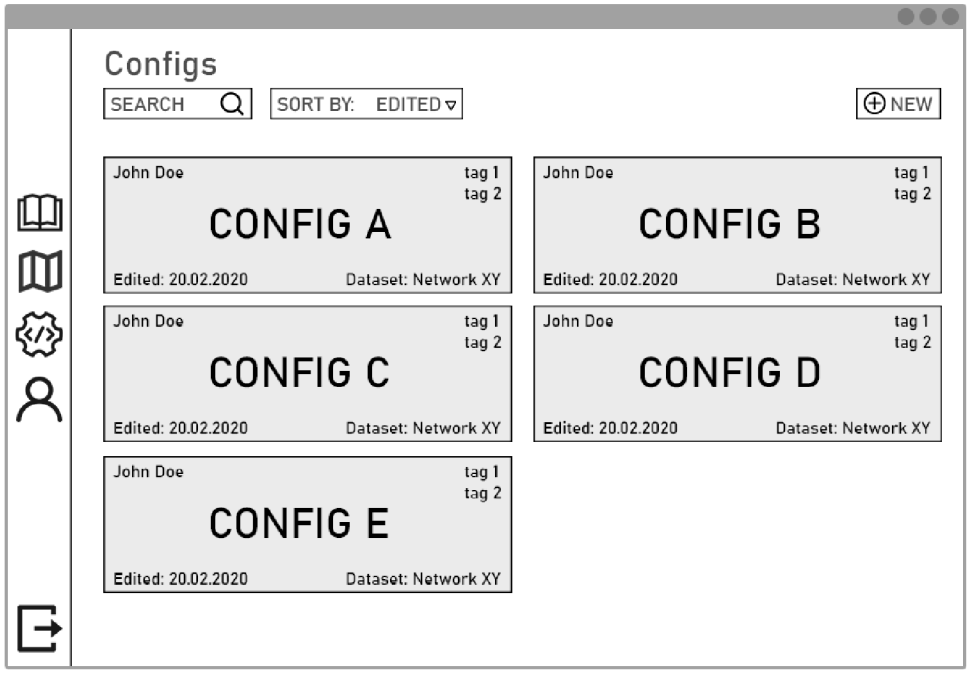
Obrázek 6.2: Návrh uživatelského rozhraní – pohled na rozpis v sekci datových sad. Zobrazeny jsou všechny přístupné datové sady včetně jejich vlastností.



Obrázek 6.3: Návrh uživatelského rozhraní – pohled na sekci datových sad při tvorbě nové datové sady. Modální okno pro proces připojení vlastního databázového systému.

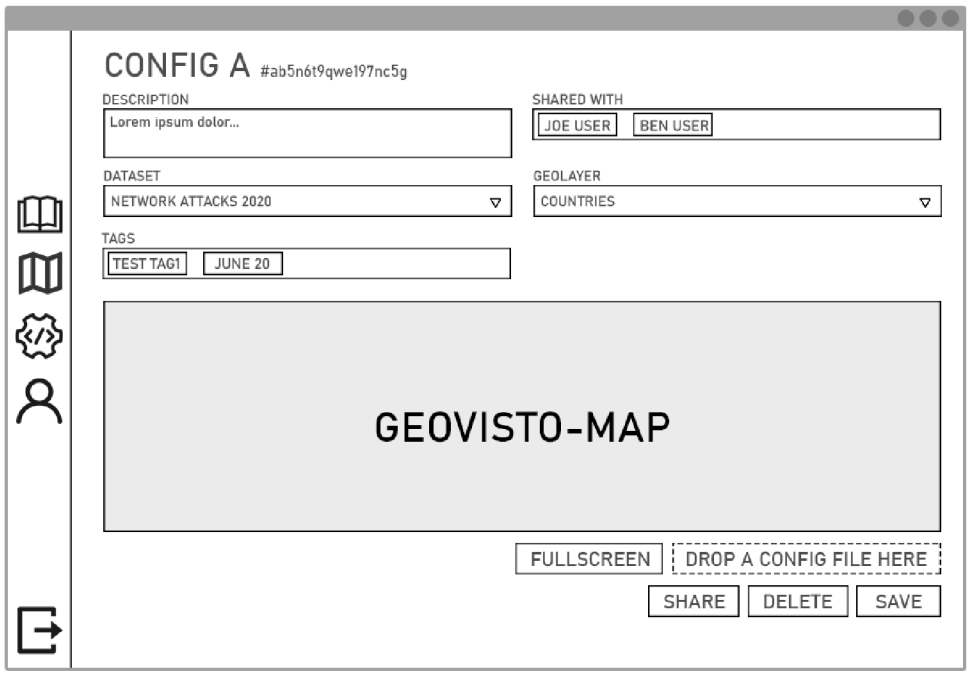
- **Sekce datových sad – detail** zobrazí stránku umožňující správu dat datové sady, čímž je myšleno změna názvu a popisu, editace samotných dat nebo nastavení oprávněni při sdílení s ostatními uživateli systému.

- **Sekce geografických vrstev** bude velmi podobná sekci datových sad. Uživatel si taktéž nejprve všimne tabulky zobrazující uložené objekty. Nebude chybět filtrace ani řazení. Pro každou z geografických vrstev bude vygenerován náhled. Dá se říct, že jediný zásadní rozdíl oproti sekci datových sad, bude chybějící možnost přidání geografické vrstvy z uživatelské databáze. Po zvolení libovolné existující geografické vrstvy se zobrazí stránka umožňující správu dat o dané vrstvě obsažených v systému.
- **Sekce konfigurací – rozpis** bude graficky zobrazovat konfigurace v podobě spravovaných vizualizací. U každé konfigurace bude ihned vidět název, vlastník, použitá datová sada a datum poslední editace. Pozadí elementů odkazujících na jednotlivé konfigurace bude vyplněno posledním známým náhledem vizualizace (obrázek 6.4). Jako bonus budou konfigurace podporovat i označení pomocí tzv. tagů (z anglického *tag*). Tvorba nové konfigurace proběhne v modálním okně výběrem datové sady a zadáním popisných atributů.



Obrázek 6.4: Návrh uživatelského rozhraní – pohled na rozpis viditelných konfigurací. Šedá plocha bude vyplněna náhledy z vizualizací.

- **Sekce konfigurací – detail**, vyobrazena v obrázku 6.5, zobrazí stránku umožňující správu dat konfigurace. To zahrnuje úpravu popisu, volbu datové sady a geografické vrstvy, volbu tagů a nastavení oprávněni při sdílení s ostatními uživateli systému. Samozřejmost je element s knihovnou Geovisto zobrazující vizualizaci včetně ovládacích prvků. Tento element bude možno zobrazit přes celou obrazovku a soustředit se tak přímo na vizualizaci. Na stránce detailu konfigurace bude také možnost získání přímého odkazu na vizualizaci pro sdílení mimo informační systém, stejně tak i předgenerovaný kód HTML elementu *iframe*.
- **Uživatelská sekce** poskytne uživatelům všechny informace o jejich profilu včetně možnosti jeho editace. Umožní změnu barevného schématu informačního systému,



Obrázek 6.5: Návrh uživatelského rozhraní – pohled na detail vytvořené konfigurace.

pomocí interaktivního grafu zobrazí zrychlený přehled všech jejich datových sad a konfigurací, včetně vazeb mezi nimi.

- **Stránka pro sdílení** zobrazí interaktivní mapu vizualizace v celé výšce a šířce okna. Vzhledem k nemožnosti anonymně upravovat jak datovou sadu, tak vlastnosti konfigurace, budou všechny ovládací prvky stránky skryté.

6.4 Databázová část

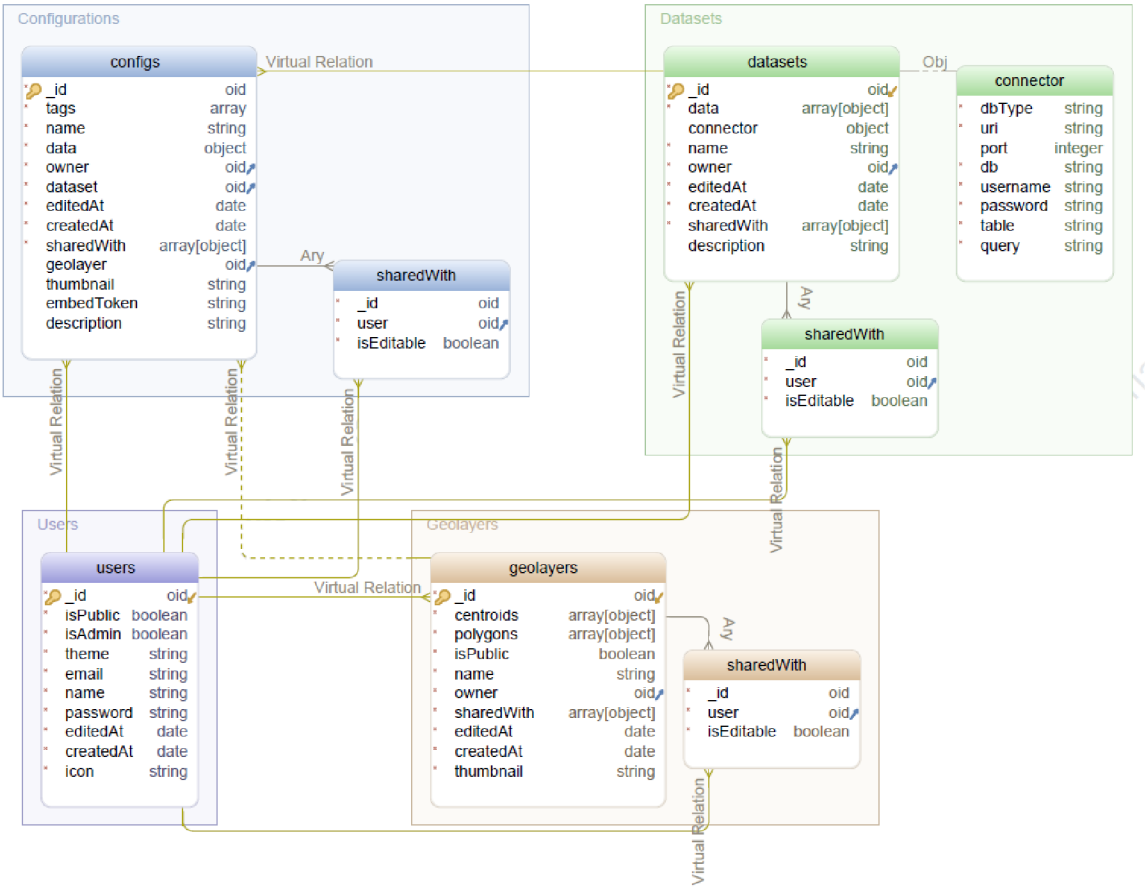
Informační systém bude pro své potřeby využívat jedinou databázi přímo připojenou k serverové části. Vzhledem k formátu ukládaných dat vyskytujících se v informačním systému, se jako rozumná volba pro databázovou část zdá některá z NoSQL databází. Například při použití dokumentově orientované databáze odpadá potřeba mapování tabulek a relací mezi nimi na entity systému.

Při uchovávání přístupových údajů uživatelských databází je potřeba uskutečnit důležité rozhodnutí z pohledu bezpečnosti, zda se tyto údaje použijí jednorázově pro přístup a budou ponechány v paměti pouze po dobu trvání aktuální relace. Data z uživatelských databází by se tak v požadovaném tvaru zduplikovala do interní databáze odkud by následně docházelo ke čtení. Takové řešení ovšem není schopna reflektovat změny dat v reálném čase.

6.4.1 Entity systému

V systému se vedle nespočtu přepravek a ad-hoc objektů budou vyskytovat 4 typy entit, které naleznou zastoupení modelem. Bude se jednat o model pro datovou sadu, konfiguraci, geografickou vrstvu a uživatele, vztahy mezi modely lze vidět na obrázku 6.6. Takový mo-

del slouží jako šablona a odpovídá způsobu uložení dat v databázi. Má přesně definované atributy včetně datových typů či specifikovaného oboru hodnot.



Obrázek 6.6: Navržené schéma databáze MongoDB znázorňující uložení entit použitých v informačním systému.

Kapitola 7

Implementace

V této kapitole se zaměřím na samotnou implementaci informačního systému pro správu vizualizací geografických dat. Opodstatním volbu daných technologií. Proniknu hlouběji do problematiky a zmíním různé způsoby implementačních detailů.

7.1 Databázová část

Vzhledem k formátu ukládaných dat, jsem se rozhodl pro NoSQL dokumentově orientovanou databázi MongoDB, která si jednoduše poradí s proměnlivou formou uchovávaných objektů. Každá z entit informačního systému, popsaných dříve v návrhu systému, bude obsahovat odpovídající kolekci dokumentů uložených v databázové části. Přestože je možné navrhnout schéma s přesně danými atributy používaných v informačním systému, entity *Config* a *Dataset* obsahují ničím neomezený atribut *data* složitého typu objekt či pole objektů, který reprezentuje data používané knihovnou Geovisto. MongoDB se zároveň těší čím dál větší popularity a v porovnání například s MSSQL si nevede vůbec špatně [13].

7.2 Serverová část

Pro výběr technologie backendové části jsem se rozhodoval mezi programovacími jazyky Python s knihovnou Django a jazykem JavaScript s knihovnou Express.js. Tyto dva kandidáty jsem vybral záměrně, protože dle zadání se bude jednat o středně velký informační systém. Vzhledem k mým dosavadním znalostem a preferencím vyhrál tento výběr JavaScript.

7.2.1 Běhové prostředí

Samotný kód jazyka JavaScript je potřeba vykonávat v běhovém prostředí, jako je například Node.js[1]. Node.js je asynchronní běhové prostředí (anglicky *runtime environment*) jazyka JavaScript řízené událostmi, navržené pro vytváření škálovatelných síťových aplikací a běžící na *enginu*[2] V8[3], který se stará například i o vykonávání kódu v jazyce JavaScript v prohlížeči Google Chrome[4]. Po každém navázaném spojení, kterých lze souběžně provádět libovolné množství, se vykoná *callback funkce*, přičemž se vývojář nemusí starat o zbylý čas

[1] https://nodejs.org/en

[2] *Engine* obvykle řeší přesně definované a výpočetně náročné úkoly a používá k tomu pečlivě optimalizované a odladěné algoritmy.

[3] https://v8.dev

[4] https://www.google.com/chrome

například aktivním čekáním na připojení. V takovém případě se Node.js jednoduše uspí [26]. Alternativou k Node.js je Deno[5], běhové prostředí pro JavaScript a Typescript také s enginem V8, které cílí hlavně na bezpečnost.

Jako kostra aplikace poslouží framework Express[6], tedy minimalistický a flexibilní framework pro Node.js, poskytující robustní sadu funkcí pro využití webovými a mobilními aplikacemi, přičemž nepřekrývá základní funkce Node.js.

7.2.2 Kooperace s databází

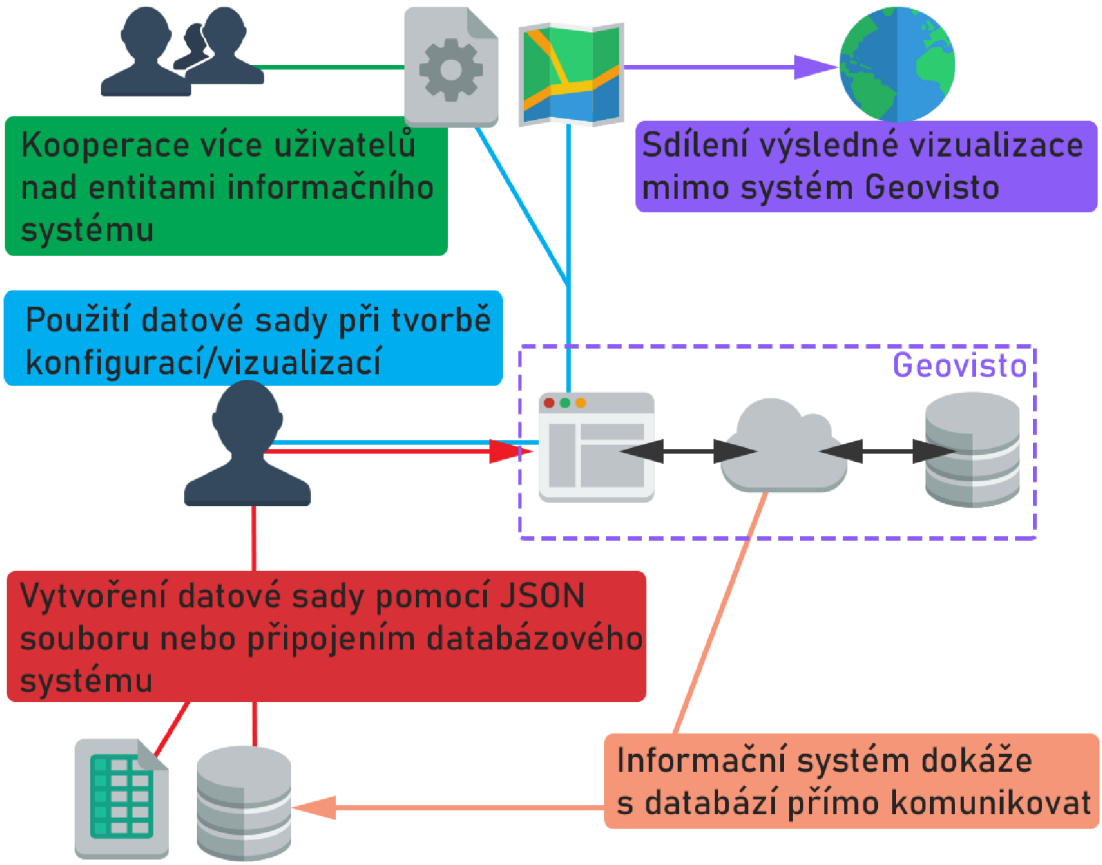
Místo klasického objektového přístupu kdy jednotlivé entity zastupují třídy, budou entity reprezentované objekty validované pomocí knihovny Mongoose[7] s definovaným datovým schématem. Mongoose se bude rovněž starat o ODM (*Object Document Mapping*), tedy mapování z objektů na dokumenty a zpět, proces vytvoření objektů a naplnění jejich atributů při získání dat z databáze informačního systému.

Jak bylo řečeno, serverová část se při správné konfiguraci pomocí klientské části naučí komunikovat s množstvím databázových systémů. Pro takové využití v prostředí Node.js existují implementace konektorů v jazyku JavaScript v podobě balíčků pro většinu z dříve zmíněných SŘBD. Z relačních databází to jsou porsager/postgres pro PostgreSQL[8], oracle/node-oracledb[9] pro Oracle Database, mysqljs/mysql[10] pro MySQL a mariadb-corporation/mariadb-connector-nodejs[11] pro MariaDB. Mezi zástupce NoSQL databází pak již zmíněný a populární konektor Automattic/mongoose [12], neo4j/neo4j-javascript-driver[13] nebo node-influx/node-influx[14]. Aktuálně systém podporuje připojení k databázím typu MySQL, PostgreSQL a MongoDB.

Pro každý z podporovaných typů databázových systémů existuje v serverové části obalovač, který obsahuje dvě hlavní funkce – *analyze()* a *getDataset()*. První zmíněná nalezne v závislosti na typu systému pro řízení báze dat seznam tabulek, kolekcí nebo čehokoliv reprezentující vnitřní strukturu. Na základě tohoto seznamu dále zjistí názvy sloupců tabulek nebo atributy objektů. V případech, kdy není pevně dané schéma jako třeba u NoSQL dokumentů v kolekci, se systém pokusí odhadnout schéma ručně. K tomu dochází tak, že z dané kolekce získá všechny objekty, následně tyto objekty sloučí a tím vytvoří jeden veliký super-objekt obsahující všechny atributy.

Uživatel si pak v uživatelském rozhraní vybere požadovanou kolekci s výčtem atributů, z nichž klientská část obratem vytvoří *query* řetězec, neboli dotaz, pro pozdější načtení dat. Tento dotaz se uloží spolu s přístupovými údaji k databázi při vytváření datové sady. Schéma zachycující kompletní použití systému včetně připojení uživatelské databáze lze vidět na obrázku 7.1.

[5] https://deno.land
[6] https://expressjs.com
[7] https://github.com/Automattic/mongoose
[8] https://github.com/porsager/postgres
[9] https://github.com/oracle/node-oracledb
[10] https://github.com/mysqljs/mysql
[11] https://github.com/mariadb-corporation/mariadb-connector-nodejs
[12] https://github.com/Automattic/mongoose
[13] https://github.com/neo4j/neo4j-javascript-driver
[14] https://github.com/node-influx/node-influx



Obrázek 7.1: Schéma zachycující práci s informačním systémem. Znázorněn je uživatel se zdroji dat, informační systém a provázání na data, kooperace uživatelů a sdílení výsledné vizualizace.

7.2.3 Záznamy o běhu systému

Po spuštění informačního systému a otevření se veřejnosti, je vhodné mít nad systémem neustále kontrolu. V případě jakékoliv nehody nebo nesrovnalosti pak zpětně zjistit, co přesně se stalo. K tomu je potřeba zaznamenávat neboli logovat všechny akce systému. K tomuto účelu poslouží knihovny Morgan[15] a Winston[16]. První zmíněný – Morgan, je *logger* HTTP požadavků v podobě *middleware* vrstvy s vlastním formátovacím jazykem a několika přednastavenými formáty. Jeho úkolem je vytvářet zprávu o každém příchozím HTTP požadavku. Winston je pak *logger* v pravém slova smyslu. Tedy univerzální nástroj podporující několik úrovní zpráv, umožňující výpis, zápis do souboru nebo dokonce odesílání zpráv na vzdálený server. Pro nasazení se tak do prostředí Express.js přidá middleware vrstva Morgan, ze které se výstup přesměruje do instance Winston.

7.2.4 Vývojové prostředí Heroku

Již během vývoje informačního systému je dobré si promyslet a vyzkoušet scénář nasazení hotového informačního systému. Pro tuto potřebu jsem si vybral platformu Heroku[17]. Heroku umožňuje nasazení, spuštění a správu aplikací napsaných v mnoha programovacích jazycích. Aby ovšem bylo splněno omezení vyplývajících z jejich bezplatného plánu, bylo potřeba změnit strukturu tak, aby informační systém místo dvou instancí Node.js (samostatná instance pro frontend a backend) vyžadoval pouze jednu. Pro tento problém jsem nalezl řešení v podobě sestavení balíčku klientské sekce, na který se následně budou přesměrovávat dotazy (detail kódu 7.1) ve skutečnosti adresované serverové části. Jediná běžící instance pak bude obsluha serverové části. V případě, kdy se nejedná o produkční verzi, chová se informační systém tak, jak byl původně navržen a umožňuje spuštění na samostatných instancích.

```
if (process.env.NODE_ENV === 'production') {
  app.use(express.static('geovisto-fe/build'))
  app.get('*', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'geovisto-fe', 'build', 'index.html'))
  })
}
```

Výpis 7.1: Detail kódu serverové části přesměrovávající požadavky do sestaveného balíčku klientské části.

7.3 Klientská část

Výběr technologie pro klientskou část značně usnadnil fakt, že aplikace Geovisto nyní vzniká v podobě komponenty frameworku React, a proto jsem se i já rozhodl použít tento framework pro tvorbu frontendu. Použití komponent umožňuje rozdělit uživatelské rozhraní na nezávislé, znovupoužitelné části a následně přemýšlet o každé části zvlášť. Zároveň, integrace takové externí komponenty je velice jednoduchá. Stačí přidat požadovaný balíček do seznamu závislostí, do požadovaného JSX souboru vložit *import statement* a komponentu v podobě nové značky (anglicky *tag*) používat. Systém pak s takovou komponentou komunikuje pomocí jejího definovaného rozhraní.

[15] https://github.com/expressjs/morgan
[16] https://github.com/winstonjs/winston
[17] https://www.heroku.com

Stejný přístup, pomocí komponent, se v případě Reactu dá použít i po vizuální stránce. Existuje množství frameworků obsahující připravené komponenty s jednotnou vizuální podobou, jako je například MATERIAL-UI[18]. Použití takového řešení ovšem ubírá vývojářům na volnosti, a proto jsem se rozhodl vybrat pro tvorbu UI klasický CSS framework poskytující více volnosti. Rozhodoval jsem se mezi v dnešní době již známými Bootstrap[19] a Foundation[20], tedy časem prověřenými *component-first* frameworky razící responzivní *mobile-first* přístup. *Mobile-first* znamená, že je již dopředu znám požadavek na využití na mobilních zařízeních, podmíněné formátování se pak využívá hlavně pro potřeby desktopových prohlížečů. Význam *component-first* pak představuje možnost tvořit výsledné uživatelské rozhraní z již vytvořených komponent obsažených ve frameworku.

Rozdílným způsobem se vydává *utility-first* framework Tailwind CSS[21]. Ten se zaměřuje na tvorbu samotných komponent pomocí tzv. primitivních utilit – tříd s již předdefinovanými styly. Touto cestou lze k daným komponentám rovnou přiřadit skupinu tříd, bez nutnosti každou třídu nejprve definovat. Třídy jsou tvořeny dodržováním jmenné konvence tak (ukázka ve výpisu 7.2), že z každého názvu lze ihned odvodit, jakým způsobem by se ke stejnému výsledku dospělo pomocí čistého CSS. Nachází se tak v něm například třídy začínající písmenem *h*, značící použití atributu *height*, s dodatkem udávající velikost – *h-0, h-0.5, h-10, h-1/2, h-screen* a podobně. Tailwind CSS tak oproti dříve jmenovaným frameworkům poskytuje o mnoho více volnosti, a proto jsem jej vybral pro použití v této práci. Ukázky uživatelského rozhraní jsou k vidění v příloze A.

```
<div class="rounded-t-md h-24 w-24 flex items-center justify-center">
    Circle
</div>

/* tailwind.css */
.h-24 { height: 6rem; }
.w-24 { width: 6rem; }
.items-center { align-items: center; }
.justify-center { justify-content: center; }
.rounded-t-md {
    border-top-left-radius: 0.375rem;
    border-top-right-radius: 0.375rem;
}
```

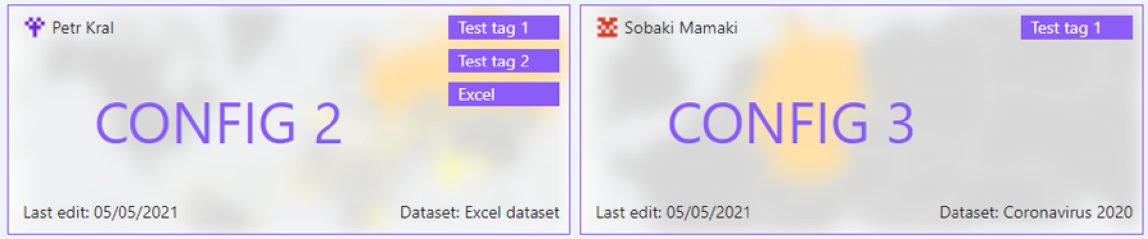
Výpis 7.2: Příklad použití několika předdefinovaných tříd frameworku Tailwind CSS (včetně vysvětlení) pro dosažení požadovaných vizuálních vlastností HTML elementu.

7.3.1 Náhledy konfigurací

Náhledy v seznamu konfigurací dodávají systému jistou míru živosti. Myšleno tak, že se nejedná o pouhé tabulkové rozpisy. Tyto náhledy (lze vidět v obrázku 7.2) jsou aktualizovány s každým uložením změny konfigurace. Přímo tak reflektují aktuální změny ve vizualizaci. K vytvoření náhledu pomáhá knihovna html-to-image[22]. Pomocí DOM reference na element

[18] https://material-ui.com/
[19] https://getbootstrap.com
[20] https://get.foundation
[21] https://tailwindcss.com
[22] https://github.com/bubkoo/html-to-image

se zobrazenou vizualizací zachytí obsah tohoto elementu v podobě SVG objektu. Od tohoto objektu pak vrátí tzv. *dataUrl*, tedy data, která jsem schopen vykreslit v elementu *canvas*, přičemž zmenším jeho dimenze na třetinu. Takto upravený náhled uložím v podobě *dataUrl* do databáze systému. Aby zmenšený náhled nepůsobil rušivě, před zobrazením v rozpisu konfigurací je na něj aplikováno rozostření pomocí kaskádových stylů.



Obrázek 7.2: Automaticky generované náhledy v rozpisu sekce konfigurací.

7.3.2 Autentizace

O autentizaci se na straně klienta stará knihovna React Auth Kit[23], fungující na principu autentizace pomocí JWT – JSON Web Tokenů. JWT je JSON objekt, který se skládá z hlavičky (*header*), dat (*payload*) a podpisu (*signature*), třech částí oddělených tečkou [19]. Cílem technologie JWT je ověření autenticity dat — tedy že data nebyla změněna během přenosu.

Po odeslání požadavku na přihlášení uživatele, získá klient z odpovědi serveru autentizační token, který byl pro uživatele vytvořen. Tento token pak použije s každým dalším požadavkem na server, díky čemuž server ověří identitu uživatele a bude pro něj schopen vykonat určité operace. Na straně serveru bude kooperovat knihovna jsonwebtoken[24], která dokáže takový token vytvořit a zakódovat.

[23] https://authkit.arkadip.me/
[24] https://github.com/auth0/node-jsonwebtoken

Kapitola 8

Testování

Testování je nedílnou součástí procesu vývoje softwaru. Provádí se za účelem získat informace o kvalitě vyvíjeného produktu, zjištění chyb a zda produkt splňuje očekávané požadavky na funkčnost. Tato kapitola je zaměřena na postup testování informačního systému a z něj vyplývající informace.

8.1 Testování funkcionality

Pro potřeby testování funkcionality jsem připravil prostředí, ve kterém byly spuštěné dvě instance Node.js, každá z nich samostatně obsluhovala jednu z hlavních částí systému, jmenovitě klient a server. Použitá databáze byla nakonfigurovaná v cloudovém prostředí MongoDB Atlas[1] a obsahovala již množinu uživatelských účtů, datových sad a konfigurací. Připraveny jsem měl 2 soubory datových sad různého rozsahu, přičemž v jedné jsem pomocí pro tento účel implementovaného skriptu upravoval hodnoty tak, aby po vizualizaci produkovaly rozdílné výsledky. Dále jsem měl připravenou instanci s technologií Docker a spuštěnými kontejnery databázových systémů MySQL, MongoDB a PostgreSQL. Tyto databáze jsem naplnil daty, které se shodují se zmíněnými soubory datových sad. Přichystaný jsem měl i konfigurační soubor vytvořený knihovnou Geovisto. Testování probíhalo takovým způsobem, že jsem prováděl požadavky plynoucí z kapitoly číslo 5. Zároveň jsem se pro ověření správnosti pokoušel napodobit scénáře dosažitelné pomocí samotné knihovny Geovisto. Tyto scénáře jsem ale v případě, kdy to bylo možné, rozšířil o přidanou funkcionalitu informačního systému a snažil se porovnat, zda jsou výsledná řešení totožná.

Při testování funkcionality, jsem mimo jiné narazil na drobné problémy uživatelského rozhraní. Jedním z nich je například pozadí modálního okna, které se v případě, kdy je obsah delší než obrazovka uživatele, nezobrazí přes celou výšku obsahu.

8.1.1 Registrace a správa uživatelského profilu

Jednalo se o ověření základní funkcionality systému. Proběhla registrace nového uživatele, při které se ověřila kontrola vstupních parametrů. Poté následovalo přihlášení. V uživatelské sekci jsem nastavil tmavý motiv systému a veřejný profil. Viditelnost profilu jsem následně zkontroloval z dalšího registrovaného účtu. Po přihlášení zpátky systém automatický nastavil správný motiv.

[1] https://www.mongodb.com/cloud/atlas

8.1.2 Správa datových sad

Pro ověření správné funkcionality správy datových sad jsem využil jak zmíněné testovací soubory včetně těch, které jsem získal použitím dříve zmíněného skriptu, tak připravené databáze. V rozpisu datových sad jsem nejprve vytvořil dva typy datových sad, první typ pomocí nahrání souboru do informačního systému, druhý typ pomocí vložení textového obsahu. Následně jsem ověřil shodu s obsahem originálního souboru pomocí funkcionality exportování datové sady ze systému s použitím utility *diff*[2].

Stejný postup ověření jsem aplikoval i při vytvoření datové sady z uživatelské databáze. Zde jsem testoval databáze ze všech tří připravených kontejnerů – MySQL, MongoDB a PostgreSQL. V první fázi jsem otestoval výběr jednotlivých dimenzí z tabulek a kolekcí. Ke konci pak výběr všech dimenzí, aby byl obsah datové sady shodný s obsahem testovacích souborů.

Otestování funkcí v rozpisu datových sad spočívalo v ověření funkčnosti prvků pro řazení výsledků a vyhledávání.

8.1.3 Správa konfigurací

Scénář pro otestování správy konfigurací byl založený na vytvoření nových konfigurací z rozdílných datových sad. Pomocí původní knihovny Geovisto jsem vytvořil vizualizaci a exportoval její konfigurační soubor. Pomocí webového uživatelského rozhraní jsem se pak snažil dosáhnout stejných vizuálních výsledků za použití tohoto konfiguračního souboru (import při tvorbě nové konfigurace nebo později při editaci již vytvořené konfigurace). Dále testování spočívalo ve změnách použitých datových sad a geografických vrstev, změny informací o uložené konfiguraci, ověření duplikace a smazání konfigurace a podobně.

Během testování exportu konfigurací jsem narazil na problém, kdy se neuložené změny nepromítly do vyexportovaného konfiguračního souboru. Takové chování, přestože není vyloženě špatné, může být pro uživatele matoucí.

8.1.4 Kooperace uživatelů

Kooperaci uživatelů jsem otestoval ve spojení s datovými sadami i konfiguracemi. Jednalo se o vytvoření menšího počtu účtů, mezi kterými jsem sdílel vytvořené entity. Tím jsem prověřil oba módy sdílení, tedy s oprávněním pro čtení i editaci.

8.1.5 Sdílení vizualizací

Scénář pro testování sdílení vizualizací vypadal následovně. Pomocí knihovny Geovisto vytvořit vizualizaci a exportovat její konfigurační soubor. Dále použít datové sady vytvořené rozdílnými způsoby, ručně aplikovat dříve provedenou konfiguraci vizualizace nebo importovat konfigurační soubor a následně vygenerovat odkaz pro sdílení. Při postupu tímto testovacím scénářem jsem narazil na chybu nedostupnosti dat, kdy se datová sada čerpá z uživatelské databáze. Chybu jsem poté opravil. Funkcionalitu sdílení jsem dále ověřil i přes vygenerovaný kód elementu *iframe*. Vizualizace sdílené těmito způsoby, byly vizuálně shodné s výsledkem dosaženým pomocí samotné knihovny Geovisto. Při testování v prostředí s jedinou instancí – v developerské verzi na Heroku, se naskytl další problém s nesprávným směrováním vygenerovaného odkazu.

[2]https://man7.org/linux/man-pages/man1/diff.1.html

8.2 Uživatelské testování

Po dokončení fáze testování funkcionality, jsem požádal o otestování systému několik přátel a členů rodiny. Nejprve jsem jim vysvětlil problematiku geografických dat včetně způsobu uložení v databázi, jak funguje vizualizace těchto dat a možné scénáře použití. Následně jsem jim poskytl informační systém s připraveným testovacím prostředím, testovací datovou sadu včetně konfiguračního souboru ve formátu JSON a přístupové údaje do jednoho z použitelných databázových systémů. Poprosil jsem je, aby zkusili provést následující scénář – zaregistrovat se do systému, vytvořit dvě datové sady obsahující totožná data. První pomocí možnosti importu ze souboru a druhou pomocí připojení databázového systému. Dále pak využít libovolnou z těchto datových sad při tvorbě konfigurace, použít poskytnutý konfigurační soubor a tím vytvořit výslednou vizualizaci. Dalším krokem pak bylo ověřit sdílení. Nejprve sdílení uvnitř systému Geovisto, tedy navzájem mezi sebou, poté i vytvořením veřejného přístupového odkazu. U tohoto testování jsem byl přímo přítomen.

Z provedeného pozorování jsem nabyl dojmu, že systém je uživatelsky přívětivý a připraven k použití. Během provádění uživatelského testování jsem nezaznamenal žádný problém jak z pohledu funkcionality tak z pohledu, že by se uživatel v systému neorientoval nebo nevěděl jak chce vyřešit daný problém.

8.3 Shrnutí

Díky provedenému testování jsem získal množinu podnětů pro budoucí vylepšení webové aplikace. Jedná se především o vyřešení maličkostí v klientské části systému, které ale neubírají systému na funkcionalitě. Mezi ně patří například úprava kaskádových stylů v místech kde aplikace nepůsobí stoprocentně konzistentně. Jako další prostor pro zlepšení jsem ve stejné části nalezl informační zprávy, které se zobrazují po vykonání určitých akcí a informují o výsledku. Zde bych rozšířil obsahovou stránku zpráv, kdy daná akce skončila neúspěchem. I přes tyto problémy proběhlo testování úspěšně. Systém tak splňuje dříve specifikované požadavky.

Kapitola 9

Závěr

Cílem této práce bylo vytvořit informační systém pro správu vizualizací geografických dat, zastřešující stávající knihovnu Geovisto. Záměrem bylo usnadnit práci s touto aplikací pomocí organizace potřebných dat nebo poskytnutím prostoru pro sdílení samotných vizualizací. V rámci teoretické části jsem analyzoval problematiku stávajících geografických informačních systémů, z čehož vyplynulo pár nedostatků i výhod, kterých jsem později mohl využít v části návrhu systému. Spuštěním informačního systému se dosáhne zpřístupnění knihovny Geovisto širšímu okruhu uživatelů, pro které byl dosavadní způsob k vytvoření vizualizace nepřekonatelný problém.

Vytvořený informační systém je nyní schopný poskytnout zázemí pro uživatele zpracovávající geografická data. Umožní jim vložit data nebo vybrat vlastní zdroj, zvolit jednu z připravených geografických vrstev nebo přidat vlastní a následně data promítnout do interaktivní mapy. Uvnitř systému mohou uživatelé sdílet jak datové sady, tak konfigurace. Výsledné vizualizace umožňuje systém sdílet i s okolním světem zpřístupněním veřejného odkazu. Tento informační systém by měl být použitelný v reálném provozu. Cíle a využití společně se systémem ve vývojové verzi byly představeny na studentské konferenci inovací, technologií a vědy Excel@FIT 2021[1].

Do budoucna je samozřejmě spousta prostoru na vylepšení. Určitě bude potřeba opravit menší problémy uživatelského rozhraní, tedy části se kterou uživatel přímo pracuje a může tak ovlivnit jeho přístup k systému jako celku. Informační systém je možno rozšiřovat například v počtu podporovaných databázových systémů nebo jej naučit zpracovávat soubory přímo z různých internetových zdrojů specifikovaných pomocí URI. Přidat podporu uživatelských skupin pro usnadnění práce v týmech. Povolit registraci pomocí účtů aplikací třetích stran, jako je Google nebo Facebook. V poslední řadě pak vylepšit administrátorskou sekci, protože s větším a větším počtem aktivních uživatelů se nevyhnutelně stane časem nepřehledná.

[1] http://excel.fit.vutbr.cz/

Literatura

[1] Geografické informační systémy, Aplikovaná kartografie - Tematické mapy. [online]. [cit. 2021-04-28]. Dostupné z: http://perchta.fit.vutbr.cz/vyuka-gis/uploads/1/gis11-mapy-rok12.pdf.

[2] *ISO 3166 - COUNTRY CODES* [online]. [cit. 2020-12-26]. Dostupné z: https://www.iso.org/iso-3166-country-codes.html.

[3] Scalable Vector Graphics (SVG) 2. [online]. [cit. 2021-04-29]. Dostupné z: https://www.w3.org/TR/SVG2/.

[4] Aronoff, S. Geographic information systems: A management perspective. *Geocarto International*. Informa UK Limited. prosinec 1989, sv. 4, č. 4.

[5] Bittinger, R. R., Fraenkel, M. L., Housel III, B. C. a Lindquist, D. B. *Client/server communication system*. Google Patents, květen 1998. US Patent 5,754,774.

[6] Burrough, P. Principles of geographical information systems for land resources assessment. *Geocarto International*. Informa UK Limited. jan 1986, sv. 1, č. 3.

[7] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S. et al. *The GeoJSON Format* [online]. Dostupné z: https://tools.ietf.org/html/rfc7946.

[8] Chang, K.-T. Geographic information system. *International Encyclopedia of Geography: People, the Earth, Environment and Technology*. Wiley Online Library. 2016, s. 1–10.

[9] Dueker, K. J. Land resource information systems: a review of fifteen years experience. *Geo-Processing (Netherlands)*. 1979.

[10] Google. *Open Location Code* [online]. [cit. 2020-12-24]. Dostupné z: https://opensource.google/projects/open-location-code.

[11] Government of Canada. *Digital Topographic Raster Maps, 1944-2012* [online]. [cit. 2020-12-12]. Dostupné z: https://open.canada.ca/data/en/dataset/d248b5be-5887-4cfb-942f-d425d82e6ea9.

[12] Guo, N., Xiong, W., Wu, Y., Chen, L. a Jing, N. A Geographic Meshing and Coding Method Based on Adaptive Hilbert-Geohash. *IEEE Access*. 2019, sv. 7, s. 39815–39825. DOI: 10.1109/ACCESS.2019.2906871.

[13] Gyorödi, C., Gyorödi, R. a Sotoc, R. A comparative study of relational and non-relational database models in a Web-based application. *International Journal of Advanced Computer Science and Applications*. Citeseer. 2015, sv. 6, č. 11, s. 78–83.

[14] Harris, R. L. *Information graphics: A comprehensive illustrated reference*. Oxford University Press, USA, 1999.

[15] HEJPETROVÁ, V. *Tématické a účelové mapy [online]*. 2013 [cit. 2021-05-01]. Dostupné z: https://theses.cz/id/gnhfh9/.

[16] Hey, A. a Bill, R. Placing dots in dot maps. *International Journal of Geographical Information Science*. Taylor & Francis. 2014, sv. 28, č. 12, s. 2417–2434.

[17] Hill, R., Hirsch, L., Lake, P. a Moshiri, S. *Guide to cloud computing: principles and practice*. Springer Science & Business Media, 2012.

[18] InfluxData. *Time series database (TSDB) explained* [online]. [cit. 2020-12-26]. Dostupné z: https://www.influxdata.com/time-series-database/.

[19] Jones, M., Bradley, J. a Sakimura, N. *JSON Web Token (JWT)* [online]. [cit. 2021-03-26]. Dostupné z: https://www.hjp.at/doc/rfc/rfc7519.html.

[20] Kachlík, J. *Grafická vizualizace geografických dat síťového provozu*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22756/.

[21] Kraak, M.-J. a Ormeling, F. *Cartography: visualization of geospatial data*. CRC Press, 2020.

[22] Maguire, D. J. An overview and definition of GIS. *Geographical information systems: Principles and applications*. Longman Scientific & Technical New York. 1991, sv. 1, s. 9–20.

[23] Masse, M. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. "O'Reilly Media, Inc.", 2011.

[24] Microsoft. *What is SaaS?* [online]. [cit. 2021-02-25]. Dostupné z: https://azure.microsoft.com/overview/what-is-saas/.

[25] Open Geospatial Consortium. *KML* [online]. [cit. 2020-12-13]. Dostupné z: https://www.ogc.org/standards/kml/.

[26] OpenJS Foundation. *About Node.js®* [online]. [cit. 2020-12-26]. Dostupné z: https://nodejs.org/en/about/.

[27] Oracle. *What a Relational Database Is* [online]. [cit. 2020-12-24]. Dostupné z: https://www.oracle.com/database/what-is-a-relational-database/.

[28] Oxford University Press. Definition of geocode [online]. In: *Lexico.com dictionary*. Dostupné z: https://www.lexico.com/en/definition/geocode.

[29] Parlament České republiky, Poslanecká sněmovna. Předpis 111/2009 Sb. 2009.

[30] Richter, R. Úvod do GIS. [online]. [cit. 2021-05-01]. Dostupné z: https://www.fi.muni.cz/usr/richter/lekceGIS/.

[31] Saylor Academy. *Essentials of Geographic Information Systems* [online]. [cit. 2020-12-23]. Dostupné z: https://saylordotorg.github.io/text_essentials-of-geographic-information-systems/s08-02-vector-data-models.html.

[32] Tejada, Z. *Non-relational data and NoSQL* [online]. [cit. 2020-12-25]. Dostupné z: https://docs.microsoft.com/azure/architecture/data-guide/big-data/non-relational-data.

[33] Wade, T., Sommer S, E. et al. *A to Z GIS, An illustrated dictionary of geographic information systems.* Esri Press, 2006.

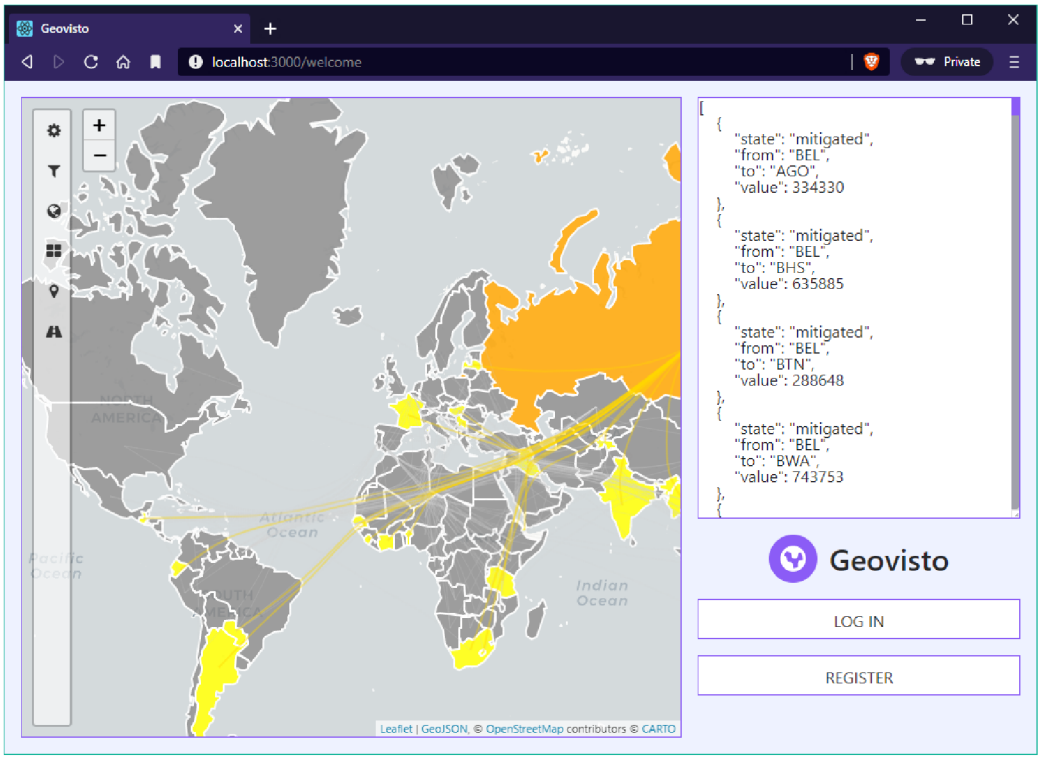
[34] Zheng, Y.-T., Zha, Z.-J. a Chua, T.-S. Research and applications on georeferenced multimedia: a survey. *Multimedia Tools and Applications.* Springer Science and Business Media LLC. nov 2010, sv. 51, č. 1.

[35] Čerba, O. Izolinie - Tematická kartografie. [online]. [cit. 2021-05-01]. Dostupné z: http://geomatika.kma.zcu.cz/studium/tka/Slides/izolinie.pdf.

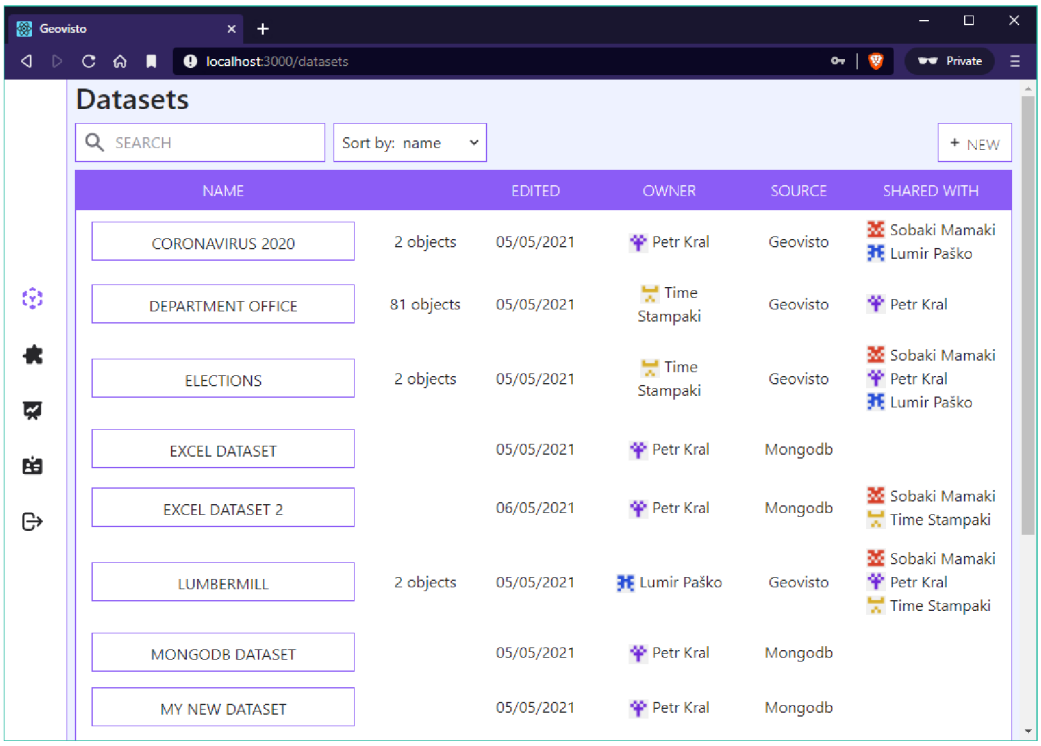
Příloha A

Snímky obrazovky

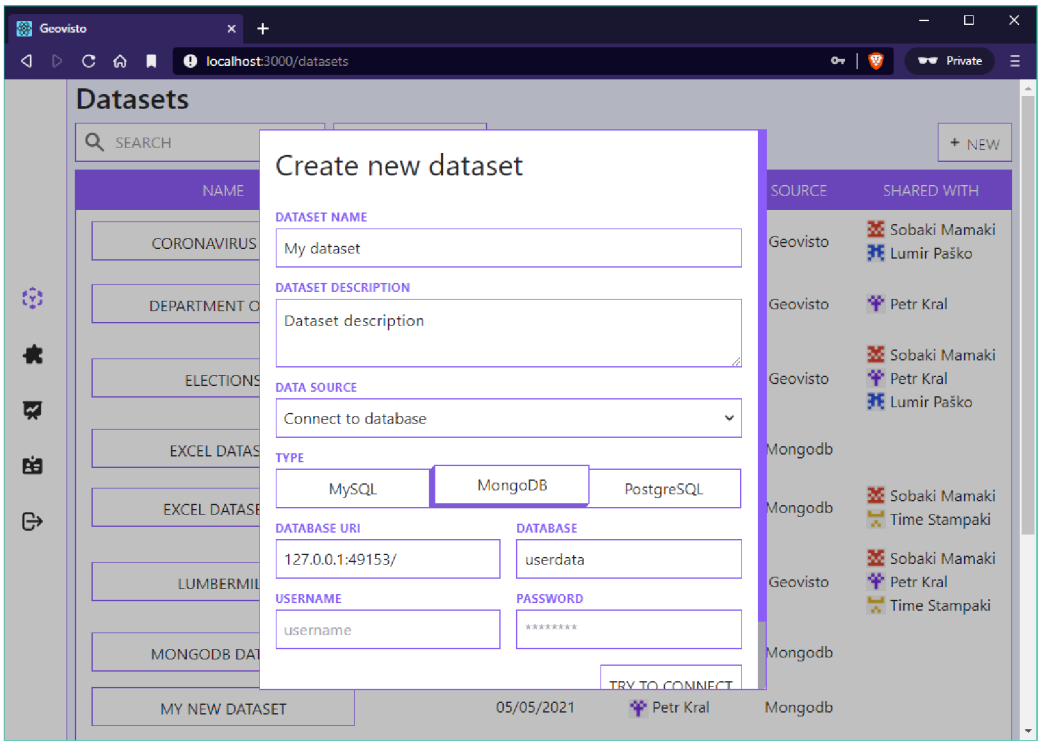
Tato příloha obsahuje množinu snímků zachycující uživatelské rozhraní informačního systému.



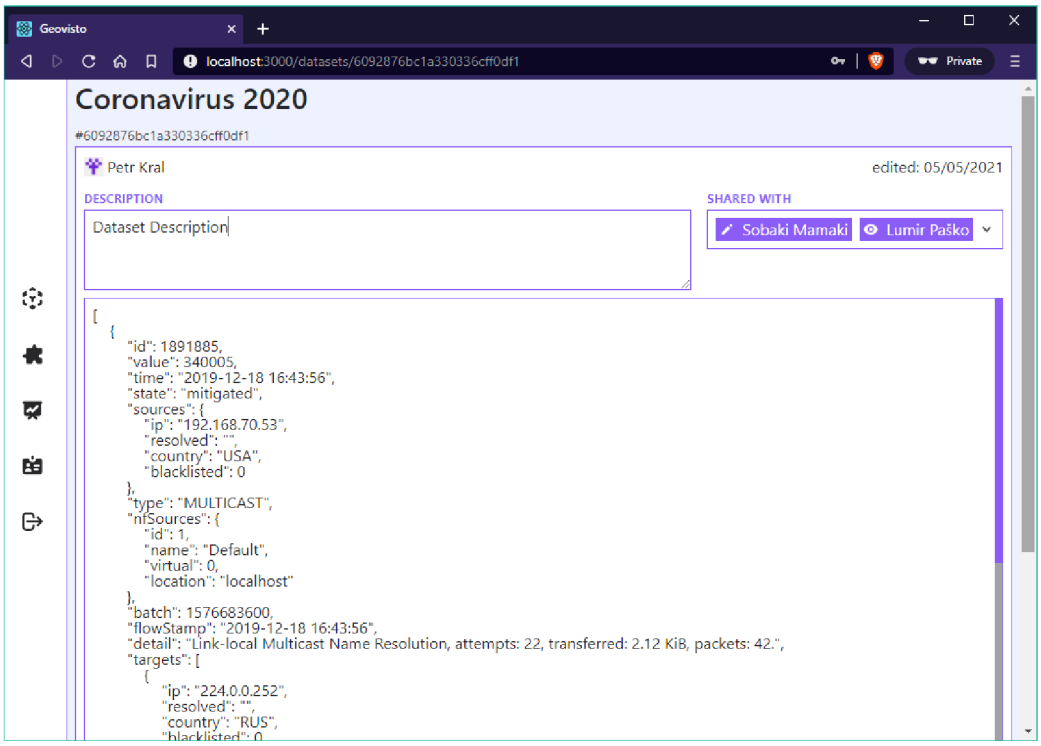
Obrázek A.1: Úvodní obrazovka obsahující oblast s komponentou pro vizualizaci a oblast představující zdroj dat. Uživatel může vyzkoušet funkcionalitu ještě před registrací do systému.



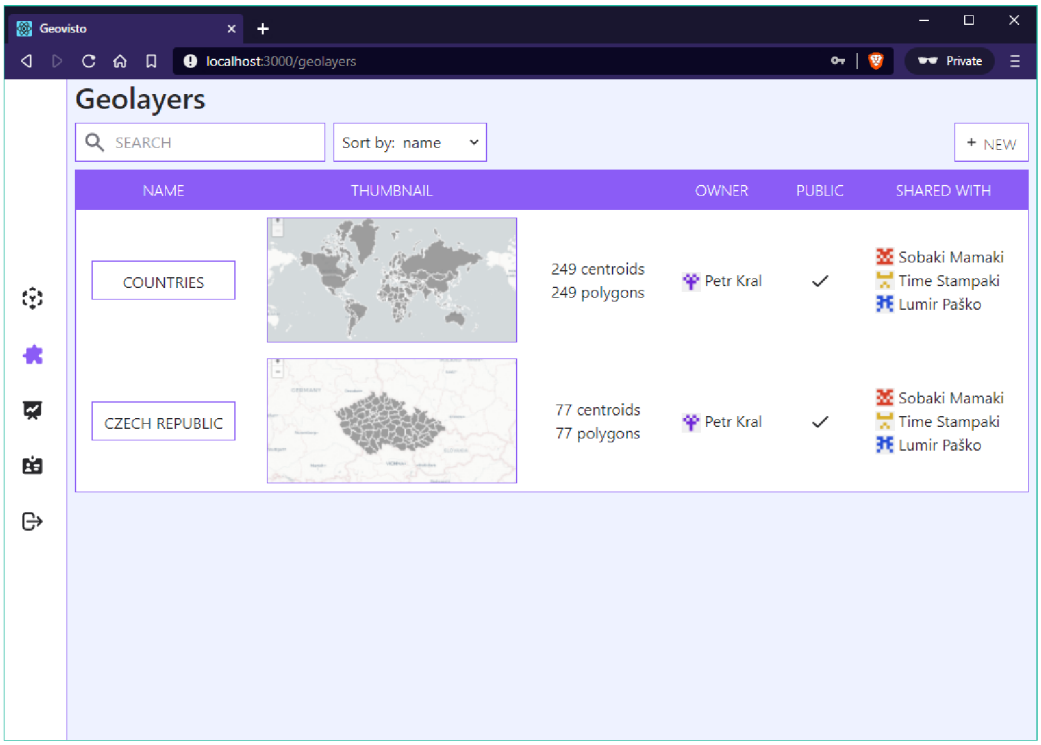
Obrázek A.2: Pohled na rozpis viditelných datových sad.



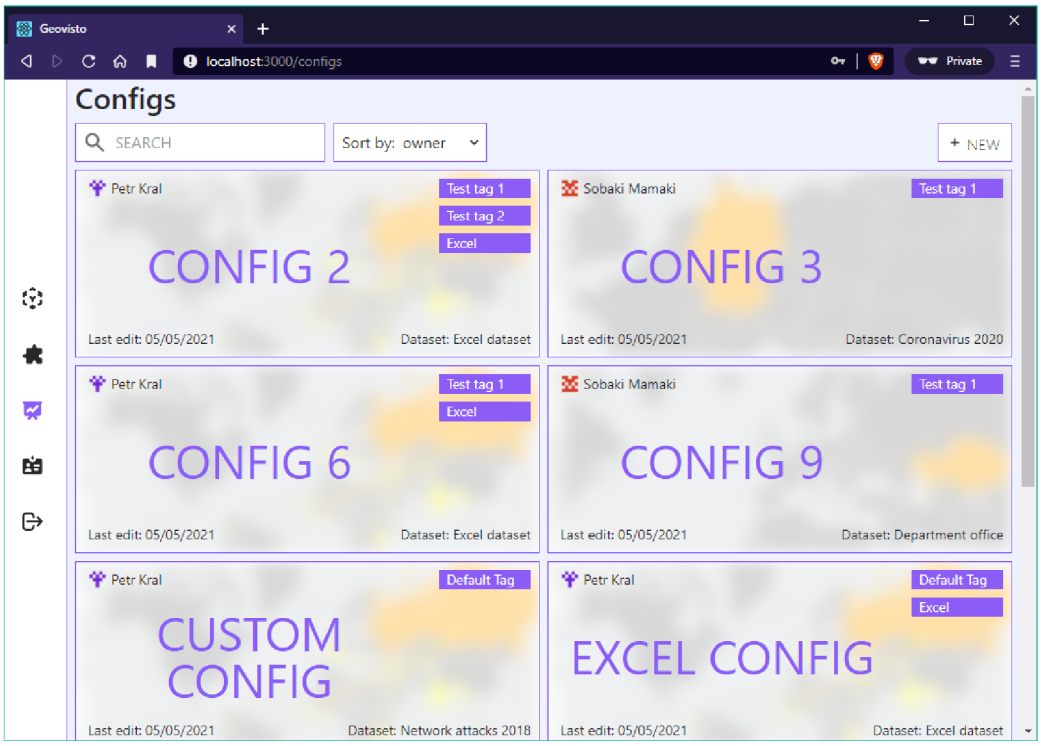
Obrázek A.3: Modální okno zobrazené při procesu tvorby nové datové sady pomocí připojení vlastního databázového systému.



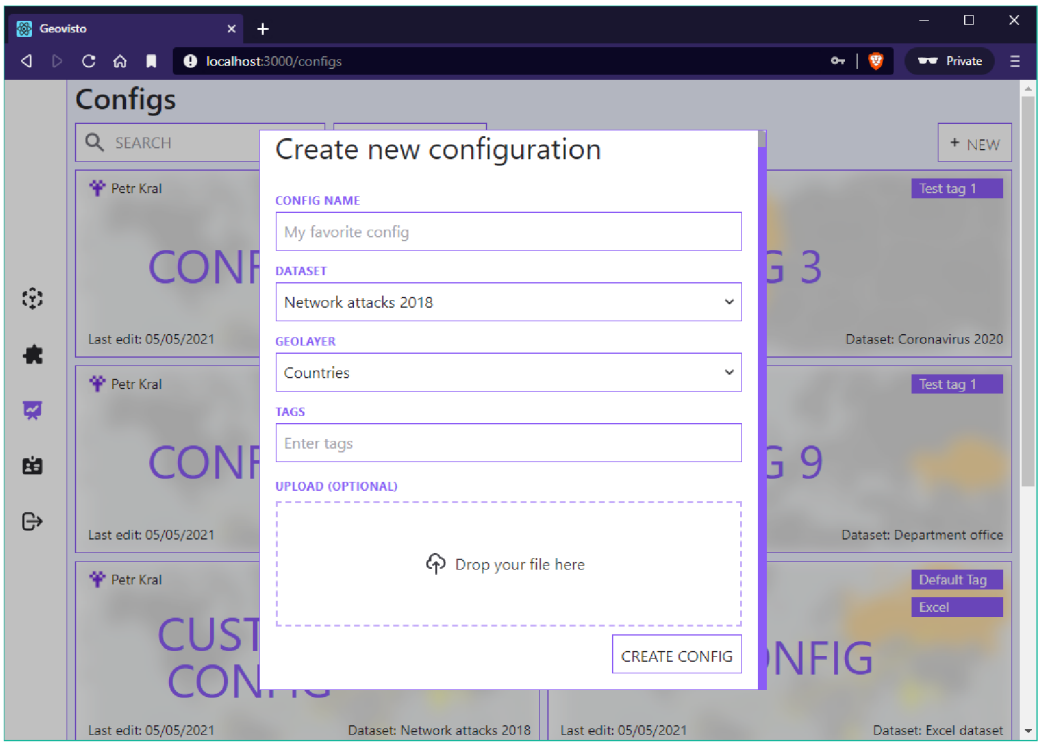
Obrázek A.4: Pohled na detail datové sady. Zobrazující popis, kooperaci mezi uživateli a surová data datové sady.



Obrázek A.5: Pohled na rozpis viditelných geografických vrstev včetně výsledku funkcionality vytvoření náhledu.



Obrázek A.6: Pohled na rozpis viditelných konfigurací. Zobrazení důležitých informací o atributech spojených s danou konfigurací.



Obrázek A.7: Modální okno zobrazené při procesu tvorby nové konfigurace.

Příloha B

Obsah přiloženého paměťového média

Tato příloha popisuje obsah přiloženého paměťového média.

V kořenovém adresáři se nachází soubor README popisující obsah média.

V adresáři *thesis* se nachází text diplomové práce ve formátu PDF a zdrojové kódy LaTeX.

Adresář *source_code* obsahuje zdrojové kódy frontendové i backendové části systému v oddělených podadresářích včetně souboru README s popisem jakým způsobem informační systém spustit.

V adresáři *attachments* je uložena informační grafika jako jsou návrhy *wireframe* nebo snímky obrazovky.

```
README
thesis
    thesis.pdf
    latex
source_code
    README
    geovisto-fe
    geovisto-be
attachments
    wireframes
    screenshots
```