České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačů

Bakalářská práce

Vizualizace GeoSPARQL dat v mapě ve webové aplikaci

Visualization of GeoSPARQL data in a web map app

Autor: Tomáš Majer

Vedoucí práce: Ing. Michal Med, Ph.D.

Studijní program: Otevřená informatika – software

Praha 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Majer** Jméno: **Tomáš** Osobní číslo: **499340**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vizualizace GeoSPARQL dat ve webové aplikaci

Název bakalářské práce anglicky:

Visualization of GeoSPARQL Data in a web map app

Pokyny pro vypracování:

1) Seznamte se s principy propojených dat, s Resource Description Framework (RDF) a standardem GeoSPARQL pro popis a dotazování prostorových dat.
2) Analyzujte knihovny pro vizualizaci prostorových dat ve webových mapových aplikacích.
3) Navrhnětě webovou aplikaci pro vizualizaci dat odpovídajících standardu GeoSPARQL v mapě, která nude umožňovat import a export dat a filtraci vizualizovaných dat pomocí SPARQL dotazů.
4) Implementujte návrh vytvořený v předchozím bodě.
5) Ověřte funkčnost a uživatelskou přívětivost nástroje uživatelským testováním (například podle System Usability Scale nebo podobně) s využitím konkrétního scénáře zaměřeného na vizualizaci, filtraci a export dat

Seznam doporučené literatury:

Open Geospatial Consortium. (2012). OGC GeoSPARQL - A Geographic Query Language for RDF Data. Retrieved from: https://www.ogc.org/standards/geosparql/
Edler, D., Vetter. M. (2019). The Simplicity of Modern Audiovisual Web Cartography: An Example with the Open-Source JavaScript Library leaflet.js. KN J. Cartogr. Geogr. Inf. 69, 51–62. Doi: https://doi.org/10.1007/s42489-019-00006-2
Battle, Robert and Kolas, Dave. (2012). Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL'. Semantic Web, vol. 3, no. 4, pp. 355-370. Doi: https://doi.org/10.3233/SW-2012-0065

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Michal Med, Ph.D. katedra počítačů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.01.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Michal Med, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....
Podpis autora práce

Poděkování

Chtěl bych poděkovat Ing. Michal Med, Ph.D. za pomoc, rady a skvělé vedení při vytváření aplikace a psaní bakalářské práce práce.

Abstrakt

Tato bakalářská práce se zaměřuje na využití principů propojených dat pro uchování geografických informací a standardu GeoSPARQL na jejich dotazování. V současné době neexistuje specializovaná aplikace umožňující prezentaci, filtraci a vyhledávání propojených prostorových dat pomocí dotazů ve výše uvedeném standardu. Cílem této práce je vytvořit jednoduchou a uživatelsky přívětivou webovou aplikaci, která umožní znázornění dat na mapě. Tato aplikace bude využívat standardizované způsoby popisu geografických dat, jako jsou linie, body a polygony, a umožní vytváření jednoduchých dat odpovídající GeoSPARQL standardu. Tento užitečný nástroj ve formě webové aplikace umožní snadnou vizualizaci a filtrování propojených dat za pomocí GeoSPARQL dotazů.

Klíčová slova

Resource Description Framework, React JS, Java, Apache Jena, Leaflet, GeoSPARQL

Abstract

This bachelor thesis focuses on the use of Linked Data principles for storing geographic information and the GeoSPARQL standard for querying it. Currently, there is no dedicated application that allows the presentation, filtering and retrieval of linked spatial data by querying. The aim of this work is to create a simple and user-friendly web application that allows the representation of this data on a map. This application will use standardized ways of describing geographic data, such as lines, points and polygons, and will allow the creation of simple data conforming to the GeoSPARQL standard. This useful tool in the form of a web application will allow easy visualization and filtering of linked data using GeoSPARQL queries.

Keywords

Resource Description Framework, React JS, Java, Apache Jena, Leaflet, GeoSPARQL

Seznam použitých zkratek

- RDF – Resource Description Framework
- W3C – Wide Web Consortium
- IRI – International Resource Identifier
- URI – Uniform Resource Identifier
- HTTP – Hypertext Transfer Protocol
- WKT – Well-Known Text
- GML – Geography Markup Language
- REST – Representational State Transfer
- API – Application Programming Interface

Seznam obrázků

Obrázek 1: Principy propojených dat [13] 11
Obrázek 2: RDF data model 12
Obrázek 3: GeoSPARQL ontology (Ontotext 2022a, p.280) 16
Obrázek 4: Ukázka REST API 20
Obrázek 5: Návrh aplikace 23

Obsah

Úvod 9
1 Sémantický web 10
2 Propojená data 10
2.1 RDF 12
2.2 SPARQL 14
2.3 GeoSPARQL 15
3 Použité technologie 19
3.1 Leaflet 19
3.2 Rest 20
3.3 Apache Jena 21
3.4 Axios 22
4 Návrh řešení 23
4.1 Nahrání a uložení dat 23
4.2 Funkcionalita klienta 24
5 Implementace 25
5.1 Nahrání dat 25
5.2 Získávání dat pomocí GeoSPARQL dotazu 25
5.3 Transformace dat na GeoJSON 27
5.4 Zobrazení na mapě 29
5.5 Vytváření nových dat 30
6 Testování 31
6.1 Charakteristika testovací skupiny 31
6.2 Testovací scénáře 31
6.3 Výsledky testování 32
7 Závěr 33
Seznam použité literatury a zdrojů 34
8 Přílohy 35

Úvod

V dnešní době se geografická data stávají stále důležitějším zdrojem informací pro různé aplikace a služby. Jedna z možností, jak tato data uchovávat a následně rozšiřovat, je využití principů propojených dat (Linked Data). Tyto principy umožňují propojení dat z různých zdrojů na webu a vytvářet bohatší a komplexnější pohled na informace.

GeoSPARQL je označení pro jazyk, ontologii a standard pro zpracování propojených geografických dat. Jedná se o rozšíření jazyka SPARQL o geografické funkce, pomocí kterých je možné využívat operace založené na poloze a geometrii. GeoSPARQL poskytuje také standardizované způsoby popisu geografických dat, jako jsou linie, body a polygony. Tyto způsoby popisu následně umožňují jednoduché porovnání a kombinování informací z různých zdrojů.

V současné době neexistuje žádná specializovaná aplikace, jež by umožnila jednoduchou a přehlednou prezentaci propojených geografických dat v mapě a zároveň umožňovala uživateli vyhledávat a filtrovat zobrazovaná data dle různých kritérií. Tento fakt komplikuje práci s prostorovými propojenými daty a omezuje možnosti jejich využití.

Cílem této bakalářské práce je vytvořit jednoduchou a uživatelsky přívětivou webovou aplikaci, která umožní vizualizaci geografických propojených dat pomocí principů práce s geometrickými daty. Tato aplikace současně nabídne možnost spouštět SPARQL dotazy nad nahranými daty v souladu s GeoSPARQL standardem.

1 Sémantický web

Sémantický web je koncept webu založený na obsahu, který je strukturován podle určitých pravidel a standardů umožňujících snadnější a efektivnější vyhledávání informací. Realizace sémantického webu předpokládá implementaci pro sémantickou, strukturální a syntaktickou složku architektury webových dokumentů. Sémantická složka se implementuje pomocí RDF (Resource Description Framework), který poskytuje standardní způsob popisu entit, vlastností a vztahů mezi nimi. V rámci strukturální části sémantického webu se používá XML (Extensible Markup language). XML slouží jako značkovací jazyk, který poskytuje hierarchickou strukturu pro organizaci a formátování dat. Umožňuje reprezentaci dat ve strojově čitelném formátu, což je konzistentní a strukturovaná prezentace informací na webu. Syntaktická část sémantického webu je zajištěna pomocí standardizovaných identifikátorů známých jako URI (Uniform Resource Identifiers). URI jednoznačně identifikují zdroje na webu a umožňují přesné odkazování a propojování dat. Slouží jako základní stavební prvek pro vytváření spojení a asociací mezi různými informacemi v rámci sémantického webu.[4][12]

```
<div vocab="https://schema.org/" typeof="Person">
       <span property="name">Jon Doe</span> was born in
       <span property="birthPlace" typeof="Place"
       href="https://www.wikidata.org/entity/Q1731">
       <span property="name">Praha</span>. </span>
</div>
```

Výpis 1.1 Příklad dat v sémantickém webu

Výsledkem aplikování uvedených standardů je konzistentní logická struktura dat, která bude implicitně vyjadřovat význam zaznamenaných informací a zajišťovat jejich strojovou čitelnost.[4]

2 Propojená data

Propojená data (Linked Data) představují sadu principů, které umožňují publikovat a sdílet strukturovaná data z různých zdrojů způsobem, jenž umožní jejich vzájemné propojení a opakované využití v rámci sémantického webu. Může se jednat jak o heterogenní systémy uvnitř jedné organizace, tak o databáze udržované různými entitami z různých částí světa.

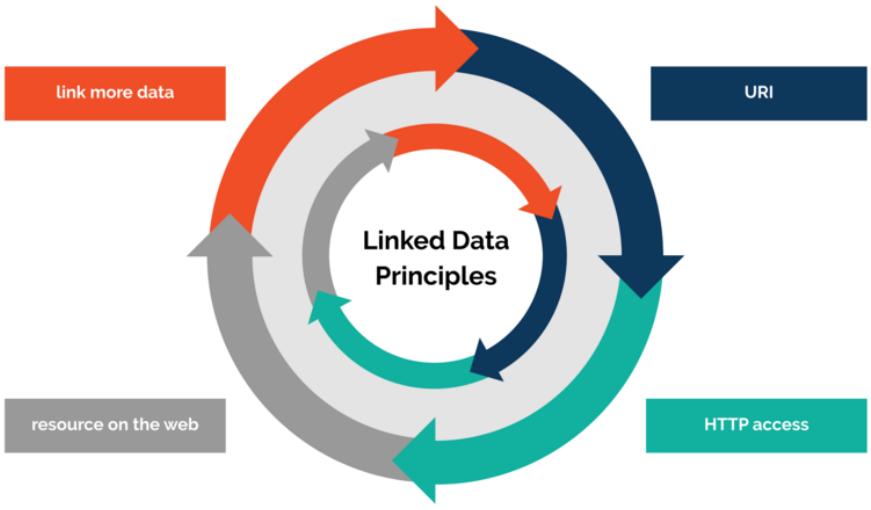
Z technického hlediska se jedná o data publikována na webu způsobem, který je strojově čitelný. Jejich význam je explicitně definován, jsou propojena na jiná data a je možné se na ně odkazovat.[1]

Propojená data se řídí zásadami uvedenými v pětihvězdičkovém plánu Tima Bernerse-Lee, který poskytuje progresivní přístup k publikování a propojování dat na webu. Každá hvězdička představuje úroveň dodržování těchto zásad. [14]

★	Zpřístupněte data na webu. [14]
★ ★	Publikujte strukturovaná data ve strojově čitelném formátu [14]
★ ★ ★	Zpřístupněte data v nechráněném otevřeném formátu [14]
★ ★ ★ ★	Používejte IRI k označení věcí, aby lidé mohli na vaše věci ukazovat. [14]
★ ★ ★ ★ ★	Propojte svá data s jinými daty, abyste získali kontext. [14]

Dodržováním pětihvězdičkového plánu propojených dat mohou vydavatelé postupně zvyšovat dostupnost a užitečnost svých dat, což povede k vytvoření propojenější a hodnotnější sítě dat. Při publikování dat na webu je třeba dodržovat čtyři jednoduché principy.

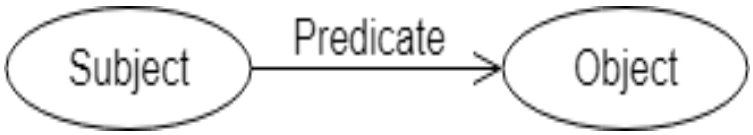
1. Používat IRI (Internationalized Resource Identifier) jako názvy věcí. [2]
2. Používat HTTP IRI, aby mohli lidé a aplikace přistupovat k těmto informacím. [2]
3. Když někdo na tuto IRI přistoupí, poskytněte data ve formě RDF nebo SPARQL. [2]
4. Přidejte do dat odkazy na jiná související data. [2]



Obrázek 1: *Principy propojených dat [13]*

2.1 RDF

RDF (*Resource Description Framework*) je framework sloužící pro reprezentaci informací a jejich výměnu na webu navržený organizací W3C (World Wide Web Consortium). RDF rozšiřuje strukturu odkazů na webu tak, že používá URI (Uniform Resource Identifier) k pojmenování vztahu mezi věcmi a také obou konců odkazu (obvykle se označuje jako "trojice"). Pomocí tohoto jednoduchého modelu umožňuje míchat strukturovaná a polostrukturovaná data, vystavovat je a sdílet v různých aplikacích.[3]



Obrázek 2: *RDF data model*

V rámci RDF jsou informace reprezentovány jako množiny trojic, z nichž každá odpovídá orientovanému grafu. Tyto trojice mají tři elementy:

- Subjekt – IRI nebo prázdný uzel
- Predikát – IRI
- Objekt – IRI, literál nebo prázdný uzel

Příkladem trojice může být věta: "*Jan se narodil v Praze*". Subjekt "*Jan*" reprezentuje zdroj informace, který je prostřednictvím predikátu "*se narodil*" propojen s objektem "*Praha*". Tato věta vytváří RDF spojení mezi osobou jménem "*Jan*" a konkrétním místem "*Praha*" vlastností "*se narodil*".

Jakýkoliv z těchto elementů může být reprezentován identifikátorem IRI, což umožňuje jednoznačnou identifikaci zdrojů. Kromě toho, že IRI fungují jako identifikátory, jsou obvykle, ale ne nutně, resolvable. To znamená, že je osoba nebo stroj může dereferencovat a dostat se k dalším informacím o zdroji. Pokud je v objektu uložena hodnota, tak se využívají literály nebo prázdné uzly.[3]

V kontextu RDF se literály používají na ukládání hodnot. Mohou být různých datových typů jako jsou například textové řetězce, čísla nebo datum a čas. Každý literál je tvořen třemi základními komponentami: hodnotou, datovým typem a jazykovou značkou. Hodnota reprezentuje samotnou informaci nebo data. Datový typ určuje, jakým způsobem je hodnota interpretována a jazyková značka v některých serializacích specifikuje jazyk, ve kterém je hodnota vyjádřena.

```
<age rdf:datatype="http://www.w3.org/2001/XMLSchema#string" xml:lang="en">13 years old</age>
```

Výpis 2.1 Příklad literálu v syntaxi RDF/XML

Na ukázce můžete vidět příklad literálu. Atribut *rdf:datatype* specifikuje datový typ a uvádí, že se jedná o řetězec pomocí URI "*http://www.w3.org/2001/XMLSchema#string*". Tím je zajištěno, že hodnota "*13 let*" je interpretována jako řetězec, nikoli jako celé číslo. Dále je zahrnut atribut *xml:lang*, který poskytuje jazykovou značku. V tomto případě je uvedena jazyková značka "*en*", která označuje, že text literálu je v angličtině.

Pro zkrácení zápisu často používaných URI se používají prefixy. Prefixy se definují na začátku dokumentu a slouží jako zkratky, které zlepšují čitelnost a přehlednost RDF dat.

RDF zahrnuje několik různých způsobů serializace (serializace je formalizovaný zápis RDF v nějakém strukturovaném formátu). Mezi hlavní používané syntaxe se řadí RDF/XML a Turtle.

Příklad dat v RDF/XML syntaxi:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://example.org/people#">
  <rdf:Description rdf:about="http://example.org/people#JohnDoe">
    <ex:name>John Doe</ex:name>
    <ex:age>30</ex:age>
    <ex:placeOfBirth>New York</ex:placeOfBirth>
  </rdf:Description>
</rdf:RDF>
```

RDF/XML syntaxe je založená na formátu XML.V tomto příkladu je použit element rdf:RDF jako kořenový prvek. Element rdf:Description definuje objekt, který popisuje osobu, a používá atribut rdf:about pro identifikaci této osoby pomocí URI. Vlastnosti osoby jsou definovány pomocí elementů ex:name, ex:age a ex:placeOfBirth, které obsahují hodnoty v textové podobě.

Příklad dat v Turtle syntaxi:

```
@prefix ex: <http://example.org/people#>.

ex:JohnDoe a ex:Person ;
    ex:name "John Doe";
    ex:age 30 ;
    ex:placeOfBirth "New York".
```

Turtle je serializace trojic za pomoci definovaných syntaktických pravidel a zkratek. Tento příklad popisuje osobu s identifikátorem "ex:JohnDoe" jako instanci třídy "ex:Person". Další řádky popisují vlastnosti osoby, jako je "ex:name", "ex:age" a "ex:placeOfBirth", a definují jejich hodnoty v textové podobě.

2.2 SPARQL

SPARQL označuje sémantický dotazovací jazyk a protokol pro data uchovaná ve formátu RDF. Byl navržen tak, aby se mohl dotazovat na širokou škálu dat a dokázal efektivně extrahovat informace skryté v nejednotných datech a uložených v různých formátech. [6]

SPARQL obsahuje čtyři typy dotazů:

1. ASK – Zeptat se, jestli existuje alespoň jedna shoda vzoru dotazu v grafu RDF.
2. SELECT – Vybrat všechny nebo některé ze shod ve formě tabulky.
3. CONSTRUCT – Vytvoří RDF graf dosazením proměnných do sady trojic.
4. DESCRIBE – Popíše nalezené shody v příslušném grafu RDF.

SPARQL dotaz se tvoří v tomto pořadí:[5]

- Deklarace prefixů pro zkracování URI nebo IRI.
  #PREFIX foo: <http://example.com/example/>
- Definice datové sady, která uvádí jaké grafy jsou dotazovány.
  #FROM..
- Klauzule o výsledku určující jaké informace mají být z dotazu vráceny.
  #SELECT..
- Vzor dotazu určující, na co se má dotazovat.
  #WHERE { ...}
- Dělení, řazení a jiné přeskupování výsledku dotazů.
  #ORDER BY ...

SPARQL má několik rozšíření. Jedním z nich je *GeoSPARQL*, který je používán pro dotazy nad propojenými geoprostorovými daty.

2.3 GeoSPARQL

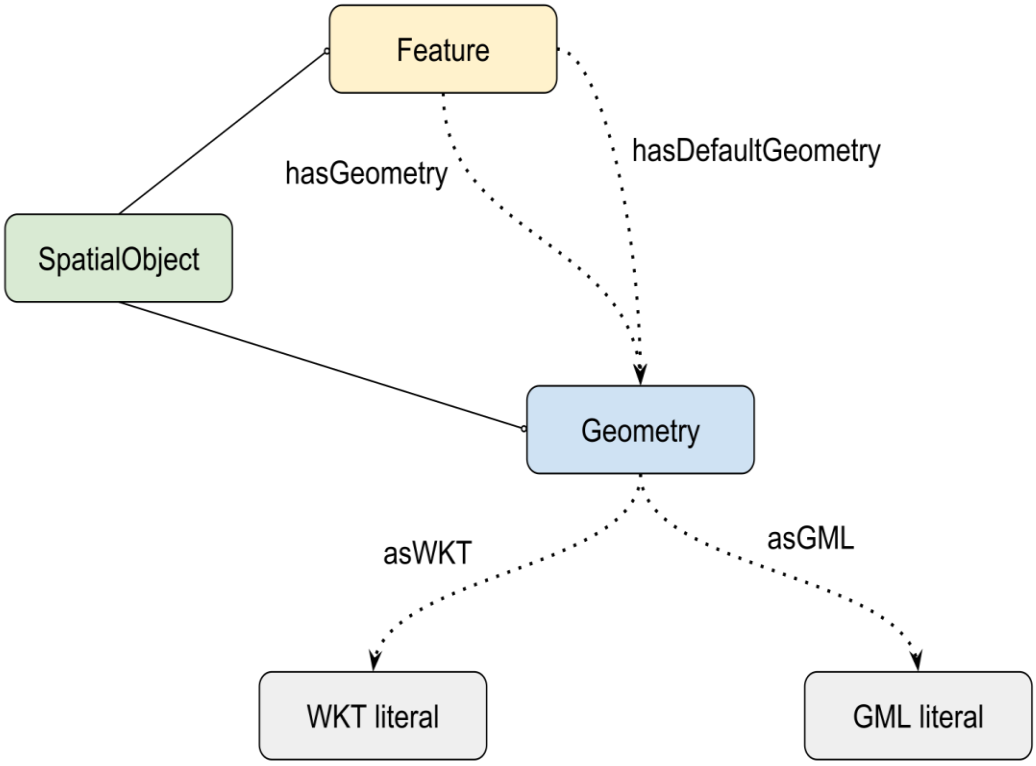
GeoSPARQL označuje jazyk, ontologii (obrázek 2) a standard pro data RDF s prostorovým rozšířením. Jedná se o rozšíření dotazovacího jazyka SPARQL, jenž je široce používán pro dotazování na data ve formátu RDF, o podporu prostorových operací. GeoSPARQL nabízí kolekci tříd, vlastností a datových typů, které jsou plně sjednocené s jednoduchými funkcemi RDF a lze je použít v rámci dotazů na grafové databáze. Dále poskytuje rozšiřující funkce jazyka pro prostorové výpočty a sadu transformačních pravidel dotazů. Díky těmto funkcím je možné efektivně manipulovat s geometrickými daty a provádět pokročilé analýzy prostorových vztahů mezi prvky datasetu. [7]

Podporovanými datovými typy pro geometrii jsou WKT (Well-Known Text) a GML (Geography Markup Language) [7]. Tyto datové typy pro geometrii jsou velmi užitečné pro zpracování prostorových dat, protože umožňují ukládání a výměnu informací o geometrii prostorových prvků, jako jsou body, linie a polygony. WKT je textový formát který umožňuje reprezentovat prvky v jednoduché čitelné formě. GML je značkovací jazyk, který vychází z XML a slouží k popisu geografických dat. Umožňuje vytvářet složitější hierarchické struktury a propojovat různé prvky do jednoho souboru.

Výchozím souřadnicovým systémem je WGS84 [7], který se používá pro globální zeměpisné sítě a zeměpisné informace. Díky tomu umožňuje přesné určení polohy bodů a jiných prvků v prostoru. V případě nutnosti může být tento systém nahrazen jiným souřadnicovým systémem, který je vhodnější pro konkrétní region nebo užití.[7]

Pro jednodušší používání, čitelnost dotazů se používají prefixy. Pro GeoSPARQL jsou nejběžněji používané prefixy "*geo*" a "*geof*". Prefix "*geo*" se používá pro zkrácení IRI "*http://opengis.net/ont/geosparql#",* která se používá k identifikaci geografických termů v ontologii GeoSPARQL. Například termín "*geo:Feature*" se používá pro označení geografického objektu a termín "*geo:hasGeometry*" se používá pro označení vlastnosti, která udává geometrii objektu. Prefix "*geof*" se používá pro zkrácení IRI "*http://www.opengis.net/def/function/geosparql/>*", která se používá pro identifikaci geografických funkcí a jejich jednoduché začlenění do dotazů.

Tyto funkce umožňují provádět geografické operace nad geometriemi. Funkce mohou být využity na výpočet vzdáleností a průsečíků mezi geometrickými objekty, transformace dat mezi různými souřadnicovými systémy, nebo zjištění vzájemných poloh objektů.[7]



Obrázek 3: *GeoSPARQL ontology (Ontotext 2022a, p.280)*

Obrázek ontologie GeoSPARQL zobrazuje hierarchickou strukturu tříd a vztahů. Klíčovými třídami jsou SpatialObject, Feature a Geometry. *SpatialObject* je jakýkoliv objekt, který má prostorový rozsah. Jedná se o nadtřídu všech prostorových objektů v tomto jazyce. *Feature* je již specifický typ prostorového objektu, který má atributy nebo vlastnosti. Může reprezentovat objekty reálného světa, jako jsou budovy, silnice nebo města a přiřazovat k nim další související informace. *Geometry* představuje konkrétní geometrický tvar prostorového objektu. Může reprezentovat body, čáry, mnohoúhelníky nebo jakýkoliv jiný geometrický tvar. Každý prvek může mít více geometrií, což naznačují dvě šipky od Feature ke Geometry na obrázku 2.

Příklad GeoSPARQL dat:

```
@prefix geo: http://www.opengis.net/ont/geosparql#
@prefix geof: <http://www.opengis.net/def/function/geosparql/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix dbo: <http://dbpedia.org/ontology>

<http://example.com/parks/central_park> a geo:Feature;
        rdfs:label "Central Park";
        geo:hasGeometry [ a geo:Geometry; geo:asWKT
        "POLYGON((-73.976974 40.785091, -73.976974 40.785091, -73.976123 40.785091, -
73.976123 40.785091, -73.976974 40.785091))"^^geo:wktLiteral ] ;
        dbo:area "3.41E8"^^xsd:double;
```

V tomto příkladu je definován jeden geografický objekt, Central Park, který je typu geo:Feature. Objekt má název "Central Park" a má geometrii, jež je reprezentována ve formátu WKT. Kromě toho se objekt odkazuje na hodnotu pro pole dbo:area, na kterém je možné pozorovat, že GeoSPARQL lze propojovat s dalšími informacemi.

Příklad dotazu na tato data:

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: http://www.opengis.net/def/function/geosparql/
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
PREFIX dbo: <http://dbpedia.org/ontology>

SELECT ?park ?label ?area
WHERE {
        ?park a geo:Feature ;
        rdfs: label ?label ;
        geo:hasGeometry/geo:asWKT ?wkt ;
        dbo:area ?area .
        FILTER(geof:sfWithin(?wkt, "POLYGON(( -74.0 40.7, -74.0 40.8, -73.9 40.8, -73.9 40.7, -
        74,0 40.7))"^^geo:wktLiteral))
}
```

Dotaz vybírá všechny geografické objekty typu geo:Feature, které jsou uvnitř zadaného polygonu (definované ve WKT formátu v sekci FILTER) a vybírá jejich název, geometrii a hodnotu dbo:area.

Výsledek dotazu by mohl vypadat takto:

park	***label***	***area***
<http://example.com/parks/central_park>	*"Central Park"*	*"3.41E8"^^xsd:double*

V tabulce by byly vidět tři sloupce: "park", "label" a "area". První sloupec obsahuje URI geografického objektu. Sloupec "label" vyjadřuje název objektu a sloupec "area" hodnotu pole dbo:area.

3 Použité technologie

3.1 Leaflet

Leaflet je populární knihovna JavaScriptu, která se využívá k vytváření interaktivních map a mapových aplikací. Jedná se o knihovnu s otevřeným zdrojovým kódem, kterou lze snadno integrovat s dalšími webovými technologiemi. Knihovna Leaflet je známá svou jednoduchostí, flexibilitou a snadným použitím, což z ní činí vhodnou volbu pro vývojáře, kteří chtějí vytvářet vysoce kvalitní mapy a geoprostorové aplikace.

Knihovna byla vytvořena v roce 2011 jako alternativa k jiným objemným mapovacím knihovnám, jako je například OpenLayers. OpenLayers je populární mapovací knihovna s mnoha funkcemi, ale často je považovaná za složitější pro začátečníky. Leaflet se na rozdíl od ní zaměřuje na jednoduchost a snadnost použití, takže je přístupnou volbou pro vývojáře všech úrovní znalostí. Knihovna je velmi malá (asi 42 kb) a nevyžaduje žádné závislosti na jiných knihovnách. [8]

Jedna z klíčových vlastností Leafletu je podpora široké škály poskytovatelů mapových vrstev, včetně OpenStreetMap, Mapbox nebo Google Maps.[8] Díky tomu mohou vývojáři do svých aplikací velmi snadno integrovat kvalitní a aktuální podkladové mapy, aniž by se museli starat o jejich ukládání, vytváření nebo správu. Všechny vytvořené mapy jsou interaktivní a mají možnost zoomovat a posouvat mapu. Leaflet dále umožňuje vytvářet vlastní body a geometrie na mapě, které se dají dále upravovat. Ke všem přidaným prvkům do mapy lze nastavit pop-up informace, které po kliknutí zobrazí dodatečné informace o objektu.

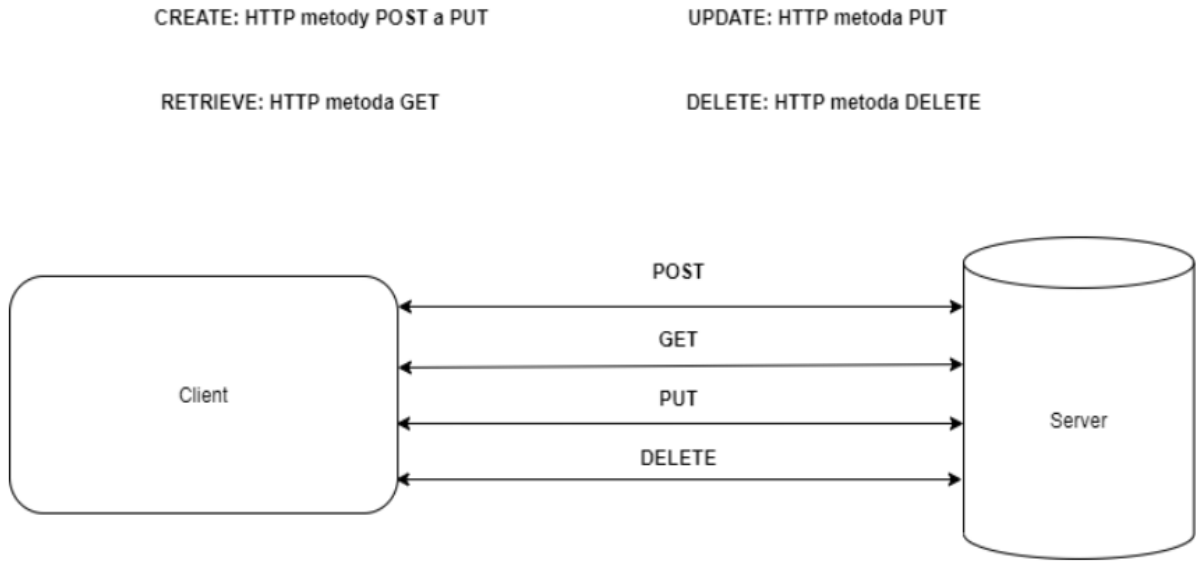
Leaflet sám o sobě neobsahuje podporu pro formát WKT ani GML pro zobrazování geometrických dat. Existuje však několik pluginů, které rozšiřují Leaflet o funkcionalitu zobrazování WKT. K tomuto účelu slouží pluginy “leaflet-wkt-parser” a “leaflet-wkt-layer”, které umožňují převádět WKT na Leaflet objekty a naopak. Umožňuje také editaci geometrií a ukládání změn do formátu WKT.

Celkově lze Leaflet považovat za dobrou volbu pro vytvoření interaktivních webových map, protože je snadno použitelný, flexibilní a rozšiřitelný. Knihovna podporuje řadu možností vlastního stylování, včetně vlastních značek, ikon a barev. Vývojáři mohou také pomocí CSS upravovat vzhled map a dalších komponent. Pro tvorbu aplikace budeme používat React verzi této knihovny, která má stejné funkce jako Leaflet.[8]

3.2 Rest

Representational State Transfer Application Programming Interface (REST API) je architektonický styl pro vytváření webových služeb, který umožňuje vzájemnou komunikaci mezi různými počítačovými systémy na internetu. Rozhraní REST API je populární a široce používaný standard pro vytváření moderních, škálovatelných a bezpečných webových aplikací. Umožňuje vývojářům vytvářet rozhraní API, ke kterým lze přistupovat přes internet pomocí standardních požadavků HTTP. [9]

Jádrem rozhraní REST API je využívání zdrojů, které jsou identifikovány pomocí jednotných identifikátorů zdrojů (URI). Webové služby, které dodržují zásady REST jsou nazývány RESTful. Typickým příkladem RESTful komunikace s protokolem http je zvládnutí operací CRUD (create,read, update, delete) nad zdroji. Metody http implementující tyto operace lze vidět na obrázku číslo 4. Díky své kompatibilitě s širokou škálou programovacích jazyků se architektura REST stala populární volbou pro tvoření moderních, škálovatelných a bezpečných webových aplikací.[9]



Obrázek 4: *Ukázka REST API*

3.3 Apache Jena

Apache Jena je open-source knihovna v jazyce Java, které poskytuje sadu nástrojů a knihoven pro vývoj aplikací pro sémantický web a propojená data. Poskytuje nástroje pro ukládání, manipulaci a dotazování na RDF data pomocí různých API, včetně SPARQL a GeoSPARQL. Apache Jena má tři hlavní funkcionality:[10]

1. Poskytuje efektivní triplestore ukládání pro RDF data.[10]
2. Umožňuje dotazování na RDF data pomocí SPARQL.[10]
3. Obsahuje nástroje pro parsování a serializaci formátu RDF.[10]

Jena ukládá informace jako RDF trojice v orientovaných grafech u umožňuje tyto informace dále přidávat, odebírat, ukládat a manipulovat. K přístupu k trojicím RDF a grafům s různými složkami se používá rozhraní RDF API. Rozhraní API podporuje přidávání a odebírání trojic do grafů a základní porovnávání vzorů grafů. Dále umožňuje načítat grafy RDF z různých externích zdrojů, souborů nebo URL, a serializovat graf ve správně formátované textové podobě. Podporuje většinu běžně používaných syntaxí RDF jako jsou RDF/XML, Turtle nebo N-triples.[10]

Typické abstrakce v tomto rozhraní API jsou:

- Zdroj představující zdroj RDF
- Literál reprezentující hodnoty dat
- Model reprezentující celý graf
- Výpis některé RDF trojice

Jednou z klíčových výhod Apache Jena je její modulární architektura. Tato knihovna je navržena tak, aby byla vysoce rozšiřitelná, s flexibilním systémem přídavných modulů, který vývojářům umožňuje snadno přizpůsobovat a rozšiřovat její funkce. Díky tomu je možné vytvářet vysoce specializované aplikace, které využívají plnou sílu technologií sémantického webu.

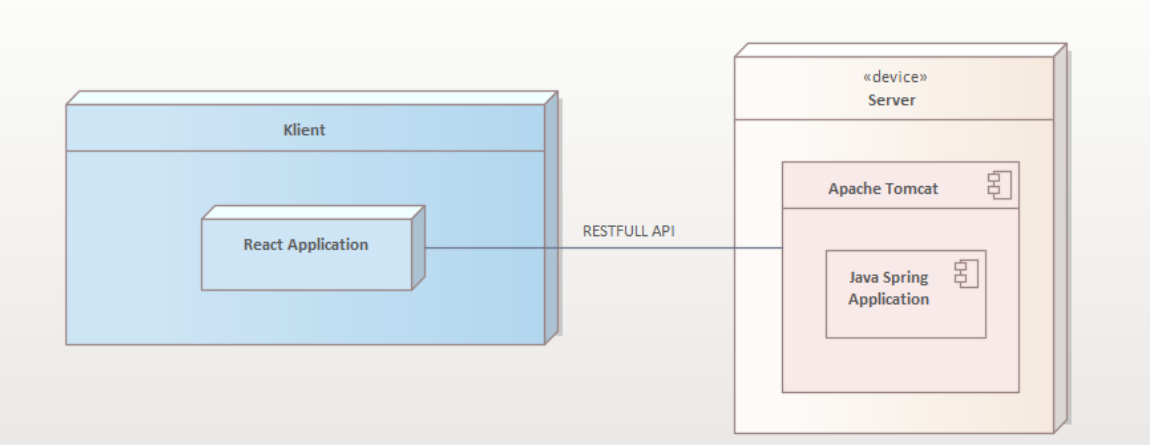
3.4 Axios

Axios je výkonná knihovna JavaScriptu, která se běžně používá k provádění požadavků HTTP z webového prohlížeče nebo serveru Node.js. Poskytuje jednoduché rozhraní API, které zjednodušuje proces odesílání asynchronních požadavků HTTP a zpracování odpovědí. [15]

Jednou z klíčových výhod Axiosu je jeho široká podpora prohlížečů. Lze jej bezproblémově používat ve všech moderních prohlížečích, což z něj činí spolehlivou volbu pro projekty vývoje webových stránek. Dále nabízí komplexní sadu funkcí, které usnadňují práci s požadavky HTTP. Podporuje různé metody požadavků, včetně GET, POST, PUT, DELETE a dalších, což vývojářům umožňuje komunikovat s rozhraními RESTful API a provádět různé typy operací. [15]

4 Návrh řešení

Tato kapitola se zaměřuje na návrh webové aplikace pro vizualizaci geografických dat v GeoSPARQL standardu, která bude umožňovat import a export dat a jejich následnou filtraci pomocí dotazů.



Obrázek 5: *Návrh aplikace*

4.1 Nahrání a uložení dat

Nahrání a uložení dat vyžaduje:

- Podpora uložení v RDF formátu a jeho serializací
- Podpora dotazů ve standardu GeoSPARQL

Pro výběr vhodného úložiště je nezbytné zohlednit několik klíčových kritérií, které mají vliv na efektivní ukládání a následné zpracování dat. První požadavek je samotná podpora RDF a jeho různých formátů. Kromě formátu RDF/XML je důležitá podpora dalších formátů. Mezi tyto formáty se mohou řadit Turtle a N-Triples. Podpora těchto formátů nám umožní snadněji načítat data z různých zdrojů a poskytne možnost výběru formátu nahrávaných dat. Další klíčovým požadavkem pro výběr vhodného úložiště je podpora pro dotazování v souladu se standardem GeoSPARQL. To nám poskytne možnost spouštět dotazy zaměřené na jejich poloze a provádět další geografické analýzy. Tyto požadavky vhodně splňuje Apache Jena.

Apache Jena je volně dostupný framework pro práci s daty v podobě RDF a dalších formátů. Jena poskytuje rychlé a robustní úložiště propojených dat a nabízí podporu pro dotazování a manipulaci s daty v GeoSPARQL standardu. Díky tomu splňuje všechny požadavky.

4.2 Funkcionalita klienta

Klient vyžaduje:

- Nahrání vlastních souborů geografických dat
- Zadávání GeoSPARQL dotazů
- Zobrazení výsledků dotazů v mapě
- Vytvoření vlastních prostorových dat
- Export dat ve formátu RDF/XML
- Konverzi WKT a GML na GeoJSON

Klient bude umožňovat nahrání vlastních souborů obsahující geografická data ve formátu RDF, a to v jednom z podporovaných formátů: RDF/XML, Turtle nebo N-Triples. Poté bude možné zadávat do aplikace GeoSPARQL vlastní nebo předdefinované dotazy a jejich výsledky zobrazit na mapě pomocí jedné z geometrických serializací. Jedná se o formáty WKT a GML, které jsou součástí GeoSPARQL standardu a umožňují popis geografických prvků na mapě pomocí bodů, linií nebo polygonů.

Pro zobrazení geografických dat na mapě bude klient využívat knihovnu React Leaflet. Tato knihovna umožňuje jednoduché zobrazení různých geometrických vrstev na mapě ve formátu GeoJSON. Při zpracování dat bude aplikace formáty WKT a GML získané z dotazu automaticky převádět do formátu GeoJSON pomocí specializovaných knihoven. Další informace o objektech zmíněné v dotazu budou také zahrnuty do informací o každém objektu a zobrazí se při kliknutí na konkrétní geometrii na mapě. Díky tomuto přístupu může uživatel snadno zobrazit data ve standardu GeoSPARQL na mapě a usnadnit jejich analýzu.

Vzhledem k tomu, že v dnešní době neexistuje příliš mnoho geografických dat odpovídající standardu GeoSPARQL, bude aplikace umožňovat vytváření jednoduchých prostorových dat za pomocí mapy nebo náhodného generování. Tímto budou moci uživatelé snadno vytvářet nové geografické entity, jako jsou body, linie nebo polygony. Vytvořená data se automaticky uloží na server a bude možné nad nimi spouštět dotazy. Všechna data se budou dát následně stáhnout ve formátu RDF/XML pro další testovací účely.

5 Implementace

Tato část se zaměřuje na klíčové prvky implementace aplikace a podrobně popisuje důležité části kódu. Implementace je představena postupně od fáze nahrávání dat až po jejich vizualizaci a spouštění dotazů nad datasetem. Na konci této části jsou popsány doplňkové funkcionality a informace o přístupu k implementaci.

5.1 Nahrání dat

První část funkcionality je nahrávání souborů na server. Data bude možné nahrávat ve formátu RDF/XML, Turtle nebo N-Tripes. Nahrávání probíhá prostřednictvím knihovny Axios, kde jsou data s obsahem nahrávaného souboru odesílána na server za pomocí Rest API. Příklad komunikace je zobrazen na ukázce *5.1*. Pokud požadavek proběhne úspěšně a data jsou na server úspěšně nahrána a přidána do Apache Jena modelu, funkce vypíše informaci o úspěšném nahrání souboru. Pokud dojde k chybě, funkce vypíše chybové hlášení do konzole a zobrazí okno s informací o chybě.

```
const handleFileUpload = (event) => {
        event.preventDefault();
        const formData = new FormData();
        formData.append("file", file);
        axios.post(serverAddress+'/upload', formData)
            .then((response) => {
                console.log(response.data);
                window.alert("File uploaded");
            })
            .catch((error) => {
                console.error(error);
                if (error.response && error.response.data) {
                    window.alert(error.response.data);
                } else {
                    window.alert("Error occurred while file upload.");
                }
            });
    };
```

Výpis 5.1: Funkce nahrávání souboru na server

5.2 Získávání dat pomocí GeoSPARQL dotazu

Pro zobrazení dat na mapě je nejdříve nutné zaslat dotaz na server. Toto probíhá s podporou technologie Axios, pomocí které se zašle GeoSPARQL dotaz na server. Dotaz se provede na serveru nad konkrétním Apache Jena modelem. Server pošle zpět odpověď ve formátu JSON *5.3*, v němž jsou uloženy výsledky vykonaného dotazu. Výsledek je na straně klienta poslán funkci *onQueryResponseDataChange*, která informuje ostatní části aplikace o změně dat.

Příklad komunikace je zobrazen na ukázce 5.2. Můžeme zde vidět poslání předdefinovaného dotazu na zobrazení všech dat, která mají geometrii a obsahují WKT nebo GML serializaci.

```
const getAllData = async (event) => {
        event.preventDefault();
        let getData= "PREFIX geo:” +
<http://www.opengis.net/ont/geosparql#>\n" +
            "SELECT *\n" +
            "WHERE {\n" +
            " ?feature geo:hasGeometry ?geometry .\n" +
            " OPTIONAL { ?geometry geo:asWKT ?wkt }\n" +
            " OPTIONAL { ?geometry geo:asGML ?gml }\n" +
            "}";
        setQuery(getData);
        try {
            const response = await axios.post(serverAddress +'/query',  {
query: getData });
            data.onQueryResponseDataChange(response.data);
            console.log(response.data);
        }  catch (error) {
            window.alert(error.response.data);
        }
    }
```

Výpis 5.2: Funkce na poslání předdefinovaného dotazu

```
{
    "head": {
        "vars": [
            "feature",
            "geometry",
            "wkt",
            "gml"
        ]
    },
    "results": {
        "bindings": [
            {
                "feature": {
                    "type": "uri",
                    "value":
"http://www.opengis.net/ont/geosparql#Feature186"
                },
                "geometry": {
                    "type": "bnode",
                    "value": "b0"
                },
                "wkt": {
                    "type": "literal",
                    "datatype":
"http://www.opengis.net/ont/geosparql#wktLiteral",
                    "value": "POLYGON((9.404296875000002
51.962433150472734,2.6367187500000004 48.942049121424716,4.306640625000001
44.17232211597221,13.271484375000002 47.299512203887055,9.404296875000002
51.962433150472734))"
                }
            }
         …
        ]
    }
}
```

Výpis 5.3: Příklad odpovědi serveru v JSON formátu

5.3 Transformace dat na GeoJSON

Pro správné zobrazení dat na mapě je nutné nejdříve analyzovat, kde se údaje o geometrii objektu nachází. K tomu je možné použít regulární výrazy, pomocí kterých se zjistí, který atribut odpovídá geometrii. Aplikace používá pro poznání WKT formátu regex: *"/^(POINT|LINESTRING|POLYGON)\s*\((.*)\)$/i"* a pro GML regex: *"/<gml:(.*?)>(.*?)<\/gml:\1>/i".* Dále je nezbytné údaje o geometrii transformovat do formátu, který je podporován knihovnou React Leaflet. Pro tuto funkcionalitu je ideální formát GeoJSON, který se dá snadno přidat do mapy pomocí vrstvy GeoJSON. Pro transformaci dat ve formátu WKT nebo GML do formátu GeoJSON je možné použít různé knihovny. V JavaScriptu lze pro konverzi z formátu WKT použít knihovnu "wellknown"[16]. Transformace se uskuteční zavoláním funkce wk.parse, která přijímá WKT řetězec jako argument a vrací geometrii ve formátu GeoJSON. Pro GML existuje knihovna "ogc-schemas"[17], která obsahuje funkce podobné. Konverze dat je zobrazena v ukázce 5.4. Nejdříve je vyhodnoceno zda je daný atribut serializací geometrie a následně se pomocí funkcí *wk.parse* nebo *gmlParser* vytvoří GeoJSON souřadnice do proměnné "geom" každého objektu.

```
const data = newData.results.bindings.map(binding => {
         const result = {};
         for (const varName in binding) {
            if (wktRegex.test(binding[varName].value)) {
                result["geom"] = wk.parse(binding[varName].value);
            }
            else if(gmlRegex.test(binding[varName].value)){
                result["geom"] = gmlParser(binding[varName].value);
            }
            else {
                result[varName] = binding[varName].value;
            }
        }
        return result;
});
```

Výpis 5.4: Transformace WKT a GML na GeoJSON souřadnice¨

Po transformaci geometrie na GeoJSON souřadnice, je nutné celý objekt přeskupit tak, aby přesně odpovídal formátu GeoJSON. Tento proces je zobrazen v ukázce 5.5. Nejprve jsou vyfiltrovány objekty, které nemají geometrii. Následně se procházejí zbylé prvky, z kterých se vytvoří nový objekt odpovídající formátu GeoJSON[5.6]. Všechny vlastnosti objektů jsou přeneseny do atributu *properties.* V tomto atributu je možné vlastnosti objektů zobrazit jako informace po prokliknuti objektu na mapě.

```
const geojsonData = data
          .filter(feature => feature.geom)
          .map(feature => {
              const properties = {};
              for (const [key, value] of Object.entries(feature)) {
                   if (key !== "geom") {
                      properties[key] = value;
                   }
              }
              return {
                   type: "Feature",
                   properties,
                   geometry: {
                        type: feature.geom.type,
                        coordinates: feature.geom.coordinates
                   }
              };
        });
```

Výpis 5.5: Přeskupení objektu na GeoJSON

```
{
   type: "Feature",
   geometry: {
      type: "LineString",
      coordinates: [
         [70.0, 75.5],
         [60.0, 65.5],
         [50.0, 55.5]
      ]
   properties :{
      prop0: "value0",
      prop1: "value1",
}
```

Výpis 5.6: Příklad formátu GeoJSON

GeoJSON obsahuje specifické pole *"geometry"*, které popisuje geometrické vlastnosti prvků. Geometrie je definována pomocí specifických typů, jako je *"Point"*, *"LineString"* a *"Polygon"* a jejich konkrétních souřadnic. Kromě geometrie obsahuje GeoJSON sekci *"properties"*, která obsahuje doplňující atributy objektu a jejich hodnoty.

5.4 Zobrazení na mapě

Zobrazení dat na mapě probíhá pomocí knihovny React Leaflet. Data v GeoJSON formátu se přidají do mapy v rámci komponenty GeoJSON [*5.6]*, která je součástí balíčku "react-leaflet". V této komponentě se nachází funkce *onEachFeature*. Tato funkce určuje, co se má stát při kliknutí na objekt na mapě. V mém případě vytvářím ke každému prvku vyskakovací okno s informacemi, které se nachází v atributu *properties*.

```
<GeoJSON
    data={geoJSON}
    onEachFeature={(feature, layer) => {
    let popupContent = '';
    for (const property in feature.properties) {
         popupContent +=
`<strong>${property}:</strong>${feature.properties[property]}<br>`;
    }
    layer.bindPopup(popupContent);
    }}
/>
```

Výpis 5.6: Vytvoření GeoJSON vrstvy v React Leaflet

5.5 Vytváření nových dat

Knihovna React Leaflet umožňuje vytváření nových objektů přímo v mapě pomocí komponenty EditControl. Nejdříve lze v části *draw* nastavit, které geometrické objekty bude možné nakreslit. Z důvodů absence několika typů geometrie ve formátu WKT, je možné vytvořit v aplikaci pouze vrstvu Marker, Polyline a Polygon. Prostřednictvím metody *onCreate* je definováno, co se má stát po vytvoření nového objektu na mapě. V mém případě je spuštěna funkce *_onCreate 5.7*. Tato funkce v závislosti na typu nového prvku vytvoří nový prvek v GeoJSON formátu a přidá ho do stávajícího seznamu vrstev. Následně je vygenerován RDF triplet odpovídající novému objektu s WKT serializací a automaticky poslán na server. Z tohoto umístění může být později stažen na jiné účely.

```
const _onCreate = e => {
        const {layerType,layer} = e;
        const {_leaflet_id} = layer;
        let rdf = "@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#> .";
        rdf += "@prefix geo: <http://www.opengis.net/ont/geosparql#>";
        rdf += `geo:Feature${_leaflet_id} rdf:type geo:Feature ;\n`;
        if(layerType === "polygon"){
                setGeoJSON(layers =>
      [...layers,{type:"Feature",properties:{id:_leaflet_id},geometry:{type
      :"Polygon", coordinates:[layer.getLatLngs()[0].map(ll => [ll.lng,
      ll.lat])]}}])
            rdf += `  geo:hasGeometry [ rdf:type geo:Polygon ;\n`;
        }
        else if(layerType === "polyline"){
                setGeoJSON(layers =>
      [...layers,{type:"Feature",properties:{id:_leaflet_id},geometry:{type
      :"Linestring", coordinates:[layer.getLatLngs().map(ll => [ll.lng,
      ll.lat])]}}])
            rdf += `  geo:hasGeometry [ rdf:type geo:Linestring ;\n`;
        }
        else if(layerType === "marker"){
                setGeoJSON(layers =>
      [...layers,{type:"Feature",properties:{id:_leaflet_id},geometry:{type
      :"Point", coordinates:[layer.getLatLng().lng,
      layer.getLatLng().lat]}}])
            rdf += `  geo:hasGeometry [ rdf:type geo:Point ;\n`;
        }
        rdf += ` geo:asWKT "${toWKT(layer,layerType)}"^^geo:wktLiteral ]`;
        const formData = new FormData();
        const file = new Blob([rdf], {type: "text/turtle"});
        formData.append("file", file);
axios.post(serverAddress+'/upload',formData).then(r => console.log(r));
    };
```

Výpis 5.6: Vytvoření nových objektů v mapě

6 Testování

V této kapitole se zaměříme na testování aplikace. Cílem testování je získat zpětnou vazbu od uživatelů ohledně funkčnosti a uživatelské přívětivosti, identifikovat případné problémy a navrhnout zlepšení.

6.1 Charakteristika testovací skupiny

Testovací skupina tvořilo pět studentů bakalářského studia informačních technologií. Studenti měli základní povědomí o konceptech propojených dat a byli seznámeni se základními principy RDF a dotazovacím jazykem SPARQL. Žádný z vybraných studentů neměl předchozí zkušenost se standardem GeoSPARQL.

6.2 Testovací scénáře

Pro účely testování byly definované různé testovací scénáře, které simulovaly různé situace a interakce uživatelů s aplikací. Byly navrženy tak, aby pokryly klíčové funkcionality aplikace a umožnily ověřit její správné fungování. Zahrnuty byly následující scénáře:

Vizualizace geografických dat

Testovací osoba prováděla kontrolu a ověřování správnosti vizualizace geografických dat ve formátu RDF na mapě. Zkoumala, zda jsou zobrazené objekty a geometrie přesné a odpovídají očekávaným hodnotám. Během testování se zaměřovala také na interaktivní funkce mapy, jako je přiblížení a posunování, a zkoumala, zda je možné zobrazit podrobnosti o objektech po kliknutí.

Testovací osoba postupovala následujícím způsobem:

1. Načtení různých datových sad.
2. Vizuální kontrola zobrazených objektů.
3. Testování interaktivních funkcí mapy.
4. Klikání na objekty na mapě a kontrolovat, zda se podrobnosti o datech zobrazují správně.
5. Zaznamenání jakýchkoli chyb, nesrovnalostí a nedostatků.

Filtrování dat pomocí GeoSPARQL dotazů

Tento scénář se zabýval filtrováním dat pomocí GeoSPARQL dotazů. Testovací osoba zadávala různé dotazy pro filtrování nahraných nebo vytvořených datových sad s cílem ověřit správné vyhodnocování dotazů.

Postup testování filtrování dat pomocí GeoSPARQL dotazů byl následující:

1. Zadání různých GeoSPARQL dotazů do aplikace.
2. Ověření, zda aplikace správně vyhodnocuje dotazy.
3. Kontrola, zda jsou filtrovaná data omezena na základě zadaných kritérií
4. Zaznamenání jakýchkoli chyb, nesrovnalostí a nedostatků.

Export dat

Testovací osoba se věnovala testování exportu dat z aplikace do formátu RDF/XML. Cílem bylo ověřit správné formátování exportovaných dat a zajištění souladu s původními daty.

Testování exportu dat do formátu RDF/XML probíhalo podle následujícího scénáře:

1. Provedení exportu dat z aplikace do formátu RDF/XML.
2. Kontrola, zda jsou exportovaná data správně formátována a splňují specifikace RDF/XML.
3. Srovnání exportovaných dat s původními daty, aby se ověřil soulad a konzistence.
4. Zaznamenání jakýchkoli chyb, nesrovnalostí a nedostatků.

6.3 Výsledky testování

Testování potvrdilo, že aplikace správně zobrazuje načtené nebo vytvořené datové sady na mapě. Interaktivní funkce, jako je přiblížení a posunování mapy, jsou plynulé a umožňují uživatelům snadno analyzovat data. Filtrování pomocí GeoSPARQL dotazů funguje správně a omezuje data na základě zadaných kritérií. Export dat do formátu RDF/XML je spolehlivý a zajišťuje dodržování formátování a soulad s původními daty.

Z testování vyplynulo, že by vzhled aplikace mohl být vylepšen, aby poskytoval uživatelům příjemnější a intuitivnější prostředí. Dalším záporem je omezený export dat do jiných formátů než RDF/XML. Aplikace momentálně neposkytuje možnost exportovat data do jiných běžných serializací RDF. Kromě toho, aplikace momentálně chybí možnost resetovat celou datovou sadu. To znamená, že uživatelé nemají k dispozici jednoduchý a rychlý způsob obnovení původního stavu aplikace nebo začít s novou datovou sadou.

7 Závěr

Hlavním cílem této bakalářské práce bylo vytvořit webovou aplikaci, která umožní vizualizaci geografických propojených dat ve standardu GeoSPARQL. Součástí aplikace je také možnost nahrání vlastních datových sad ve formátu RDF a jejich následné filtrování pomocí GeoSPARQL dotazů.

V první části práce byl popsán a vysvětlen princip propojených dat a standardu GeoSPARQL pro jejich popis a dotazování. Výsledkem bylo ověření, že lze tyto koncepty úspěšně aplikovat na jejich vizualizaci v mapě za pomocí geometrické serializace WKT nebo GML.

Druhá část práce byla zaměřena na analýzu a výběr vhodných technologií a frameworků pro implementaci webové aplikace. Po detailní analýze nástrojů byly vybrány technologie React Leaflet, Apache Jena a REST API jako optimální kombinace pro dosažení stanovených cílů. Pro část klienta byl zvolen framework React s knihovnou React Leaflet, která umožňuje jednoduchou tvorbu interaktivních map. Pro serverovou část byla zvolena knihovna Apache Jena, která umožňuje jednoduché zpracování, uložení a dotazování na data v souladu s GeoSPARQL standardem. Pro vzájemnou komunikaci bylo vybráno rozhraní REST API, které umožňuje jednoduchou výměnu dat mezi klientem a serverem.

Po výběru a analýze vhodných technologií následuje popis implementace, během které byla vytvořena webová aplikace. Aplikace umožňuje nahrání vlastních prostorových datových sad a jejich následnou filtraci pomocí GeoSPARQL dotazů. Vzhledem k nedostatku dat v tomto standardu, byly přidány funkcionality na vytvoření vlastních jednoduchých dat a jejich následné stažení ve formátu RDF/XML.

Po dokončení implementace bylo provedeno testování aplikace, které se zaměřilo na funkčnost a přívětivost uživatelského rozhraní. Během testování byly prováděny různé testovací scénáře orientované na vizualizaci, filtraci a export dat.

Zadání práce považuji za splněné. Všechny požadavky aplikace byly implementovány a výsledná aplikace otestována a připravena k použití.

Seznam použité literatury a zdrojů

1. BIZER, Christian, Tom HEATH a Tim BERNERS-LEE. *Linked Data-The Story So Far* [online]. [cit. 2023-01-10]. Dostupné z: http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf
2. BERNERS-LEE, Tim. *Linked Data* [online]. [cit. 2023-01-10]. Dostupné z: https://www.w3.org/DesignIssues/LinkedData.html
3. CYGANIAK, Richard, David WOOD a Markus LANTHALER. *RDF Concepts and Abstract Syntax* [online]. [cit. 2023-01-10]. Dostupné z: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
4. *Databáze Národní knihovny ČR* [online]. [cit. 2023-01-10]. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000000556&local_base=KTD
5. *GeoSPARQL* [online]. [cit. 2023-01-12]. Dostupné z: https://opengeospatial.github.io/ogc-geosparql/geosparql11/spec.html
6. *What is SPARQL* [online]. [cit. 2023-01-13]. Dostupné z: https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/
7. *GeoSPARQL - A Geographic Query Language for RDF Data* [online]. [cit. 2023-01-14]. Dostupné z: https://www.ogc.org/standards/geosparql/
8. *Leaflet* [online]. [cit. 2023-01-15]. Dostupné z: https://leafletjs.com/index.html
9. *What is REST API?* [online]. [cit. 2023-01-20]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api
10. *Apache Jena* [online]. [cit. 2023-01-20]. Dostupné z: https://jena.apache.org/index.html
11. *RDF 1.1 Turtle* [online]. [cit. 2023-05-10]. Dostupné z: https://www.w3.org/TR/turtle/#language-features
12. LAWTON, George. *Semantic Web* [online]. [cit. 2023-05-19]. Dostupné z: https://www.techtarget.com/searchcio/definition/Semantic-Web
13. THIERY, Florian. *Linked Data Principles* [online]. In: . [cit. 2023-05-19]. Dostupné z: https://commons.wikimedia.org/wiki/File:Linked_Data_Principles.png
14. HAUSENBLAS, Michael. *5 ★ OPEN DATA* [online]. [cit. 2023-05-19]. Dostupné z: https://5stardata.info/en/
15. *Axios* [online]. [cit. 2023-05-21]. Dostupné z: https://axios-http.com/docs/intro
16. *Wellknown library* [online]. [cit. 2023-05-21]. Dostupné z: https://www.npmjs.com/package/wellknown
17. *Ogc-parser library* [online]. [cit. 2023-05-21]. Dostupné z: https://www.npmjs.com/package/ogc-parser

8 Přílohy

Github repositář:

- Zdrojové kódy: *https://gitlab.fel.cvut.cz/majerto4/GeoSparqlApp*
- Uživatelské rozhraní aplikace: *https://gitlab.fel.cvut.cz/majerto4/GeoSparqlApp/-/tree/master/src/main/app*