FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

SPARQL Endpoint a interaktivní vizualizace dat

František Kaiser

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

SPARQL Endpoint a interaktivní vizualizace dat

Bc. František Kaiser

Vedoucí práce
Ing. Petr Včelák, Ph.D.

PLZEŇ 2024

Citace v seznamu literatury:
KAISER, František. `SPARQL Endpoint a interaktivní vizualizace dat`. Plzeň, 2024. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Petr Včelák, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. František KAISER**
Osobní číslo:	**A22N0050P**
Studijní program:	**N0613A140040 Softwarové a informační systémy**
Téma práce:	**SPARQL Endpoint a interaktivní vizualizace dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s metodami analýzy a vizualizace datových sad a možnostmi SPARQL Endpointu.
2. Analyzujte dostupné nástroje analýzy a vizualizace dat.
3. Navrhněte vhodné metody a způsoby podpory vizualizace dat ze SPARQL Endpointů. Zaměřte se na možné datové typy včetně URI, hodnoty číselníků/kategorie i možnost předzpracování výsledku dotazu před samotnou vizualizací.
4. Implementujte navržené metody pro interaktivní vizualizaci dat včetně možnosti si předem definovat formáty a pohledy dle typu dat.
5. Důkladně otestujte a také proveďte validaci na netriviální datové sadě.
6. Proveďte kritické zhodnocení a diskuzi dosažených výsledků.

Rozsah diplomové práce: **doporuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Ing. Petr Včelák, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **8. září 2023**
Termín odevzdání diplomové práce: **16. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2023

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 14. května 2024

. . . . . .
František Kaiser

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Diplomová práce se zabývá podporou analýzy a vizualizace propojených dat ze SPARQL Endpointu. Popisuje sémantické technologie, procesy analýzy a vizualizace dat. Jsou porovnány nástroje, které problematiku řeší. Je navržena a implementována aplikace s architekturou klient-server, která umí pomocí automatických i manuálních transformací zpracovat propojená data v surové podobě do přehledných tabulek. Sloupce jsou formátovány dle datového typu nebo dalších požadavků uživatele a jsou doplněny o statistické přehledy složení hodnot a jejich typů, které lze interaktivně filtrovat. Data lze dále prezentovat formou vícestránkových dashboardů, které obsahují několik typů vizuálů. Implementace je testována jednotkovými testy, obrazovky klienta jsou testovány manuálně. Řešení je validováno na netriviálních dotazech, je změřena výkonnost, a vzniklá aplikace je porovnána s analyzovanými nástroji. Nakonec jsou popsány možnosti optimalizace a vylepšení, kterými může být aplikace rozvíjena.

Abstract

Master thesis aims at designing methods of analysis and visualization for linked data from SPARQL Endpoint. Semantic technologies, data analysis and visualization processes are described, existing tools are compared and an application with a client-server architecture is designed and implemented. Application processes linked data in raw form into well-arranged tables, using automatic and manual transformations. Columns are formatted according to data type or other user requirements, they provide statistical summaries of their values and types, which can be interactively filtered. Data can be further presented in the form of multi-page dashboards that contain several types of visuals. Implementation is unit tested and client screens are tested manually. Solution is validated on non-trivial queries, performance is measured and result compared with analyzed tools. Finally, optimization and enhancement possibilities are described.

Klíčová slova

RDF • OWL • SPARQL • SPARQL Endpoint • interaktivní vizualizace • zpracování SPARQL • analýza SPARQL • analýza RDF dat

Poděkování

Rád bych tímto poděkoval vedoucímu diplomové práce Ing. Petru Včelákovi, Ph.D. za vedení, užitečné připomínky a čas strávený konzultacemi.

Obsah

Úvod 1

Propojená data a sémantické technologie poskytují rozdílné možnosti zpracování dat než např. data uchovaná v relačních databázích. V tomto kontextu se práce zabývá průzkumem existující podpory takových datových sad v relevantních aplikacích analýzy a vizualizace dat. Plně využité možnosti sémantických technologií a SPARQL endpointů mohou efektivně sloužit k získání relevantních metadat, což u běžných dat není přímočaře možné. Jejich reálná použitelnost musí být prakticky ověřena.

Cílem této práce je na základě analýzy možností sémantických dat a SPARQL Endpointu navrhnout, realizovat a ověřit vhodné metody podpory zpracování propojených dat a jejich vizualizace. Z poznatků analytické části vznikne návrh metody pro celý proces analýzy a vizualizace dat, a bude definována specifikace požadavků na SW. Budou zvoleny vhodné technologie, pomocí nichž bude implementována aplikace, která specifikované požadavky řeší. Implementovaná aplikace prakticky ověří vhodnost navržených metod, řešení bude důkladně otestováno a zvalidováno.

Diplomová práce v teoretické části zkoumá propojená data, sémantické technologie, RDF formáty, slovníky, ontologie a možnosti SPARQL endpointu. Následně jsou rozděleny procesy metod analýzy a vizualizace dat, které jsou blíže popsány. Další část analyzuje a porovnává jednotlivé procesy u relevantních aplikací datové analýzy a vizualizace dat: Power BI, Tableau a Looker Studio. Na základě analýzy je navržena metoda podpory definovaných procesů pro propojená data a jsou specifikovány požadavky na SW. Pro realizaci seznamu požadavků je navržena aplikační databáze a jsou zvoleny vhodné technologie. V praktické části následuje popsání realizované implementace a popis důkladného otestování jednotkovými testy a manuálním testováním. Nakonec je validována výkonnost realizovaného nástroje a splnění specifikovaných požadavků. V diskuzi je řešení vyhodnoceno a porovnáno s analyzovanými aplikacemi, jsou popsána možná omezení a možnosti, kterými může být aplikace dále rozvíjena.

Propojená data a sémantické technologie

2

Nejprve budou popsány principy sémantických technologií a datový model *RDF*, jeho formáty a sémantická rozšíření.

2.1 Sémantické technologie

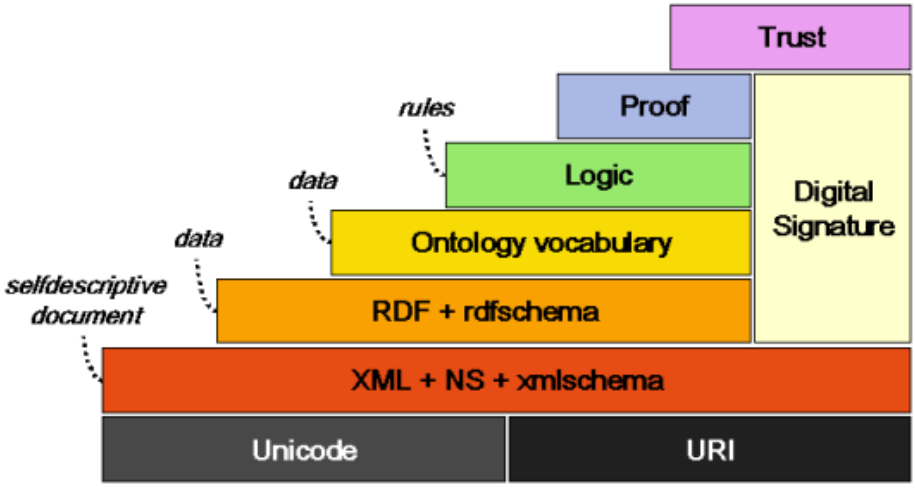
Propojená data představují vizi, kde jsou všechna data vzájemně provázaná a obsahují i sémantickou informaci, což znamená, že je jasně přiřazen jejich význam. Tato konceptualizace známá jako *Linked Data* má za cíl vytvořit globální databázi, se kterou mohou stroje lépe pracovat. K dosažení tohoto cíle jsou zavedeny sémantické technologie, jako jsou *RDF, SPARQL, OWL* a další. Na technologie, které jsou pro tuto práci podstatné, se blíže podíváme později v textu. [1]

Dle [1] jsou technologie propojených dat postaveny na šesti klíčových principech:

- *Identifikovatelnost pomocí URI:* Všechny entity, včetně lidí, míst a věcí ve fyzickém světě, mají svůj jedinečný identifikátor v podobě *URI* (Uniform Resource Identifier). To umožňuje jednoznačné identifikování konkrétního objektu zdroje dat.
- *Typy pro zdroje a vazby:* Zdroje a jejich vzájemné vazby jsou typovány, což umožňuje počítačům lépe interpretovat jejich význam. Typy přiřazují význam jednotlivým prvkům.
- *Tolerance k neúplným informacím:* Nejsou tolerovány neúplné informace. Vazby na neexistující zdroje neznamenají selhání, pouze nenalezení daného zdroje.
- *Nepotřeba absolutní pravdivosti:* Všechny informace nejsou vždy absolutně pravdivé. Aplikace, které zpracovávají dané informace, jsou zodpovědné za zhodnocení jejich pravdivosti.

- *Podpora pro evoluci:* Sémantické technologie podporují evoluci definic a konceptů. Různá uskupení mohou definovat podobné koncepty, řešit nejasnosti a nekonzistence.
- *Minimalistický design:* Sémantické technologie jsou navrženy s ohledem na jednoduchost. Standardizují se pouze nezbytně nutné prvky a na těchto základech jsou budovány jednoduché aplikace.

Obrázek 2.1 zobrazuje základní vrstvení sémantických jazyků. Jazyky jsou postaveny na základní infrastruktuře *World Wide Webu*, kterou tvoří adresace *URI*[1] a protokol *HTTP*. Tyto jazyky respektují rozmanitost požadavků na reprezentaci znalostí. Proto jsou tyto jazyky strukturovány do vrstev, kde každá vrstva poskytuje odlišnou úroveň vyjadřovací schopnosti v kontextu znalostní reprezentace. [3]



Obrázek 2.1: Vrstvy sémantických jazyků [3]

Nejnižší vrstva představuje vrstvu syntaktické reprezentace, která je zastoupena například jazykem *XML*. Vrstva *RDF (viz část 2.2)* umožňuje definovat libovolné vztahy mezi objekty nebo jejich kategoriemi bez explicitní specifikace významu vazeb mezi objekty. *RDF Schema (RDFS viz kapitola 2.4.2)* poskytuje minimální sémantickou vyjadřovací schopnost, definuje význam některých prvků, zejména tříd *(Class)* a taxonomických vztahů podtříd *(subClassOf)*. [3]

Vrstva *Web Ontology Language (OWL)* poskytuje pokročilou reprezentaci znalostí na úrovni deskripční logiky. Následují vrstvy pravidel, které poskytují vyjadřovací schopností na úrovni procedurálních znalostí. [3]

Rozdělení sémantického webu na vrstvy má důležitý význam. Vyjadřovací schopnost vyšší vrstvy v zásadě zahrnuje i vyjadřovací schopnost vrstvy nižší (platí pro

[1] `URI - Uniform Resource Identifier` je formát unikátního označení zdrojů [2]

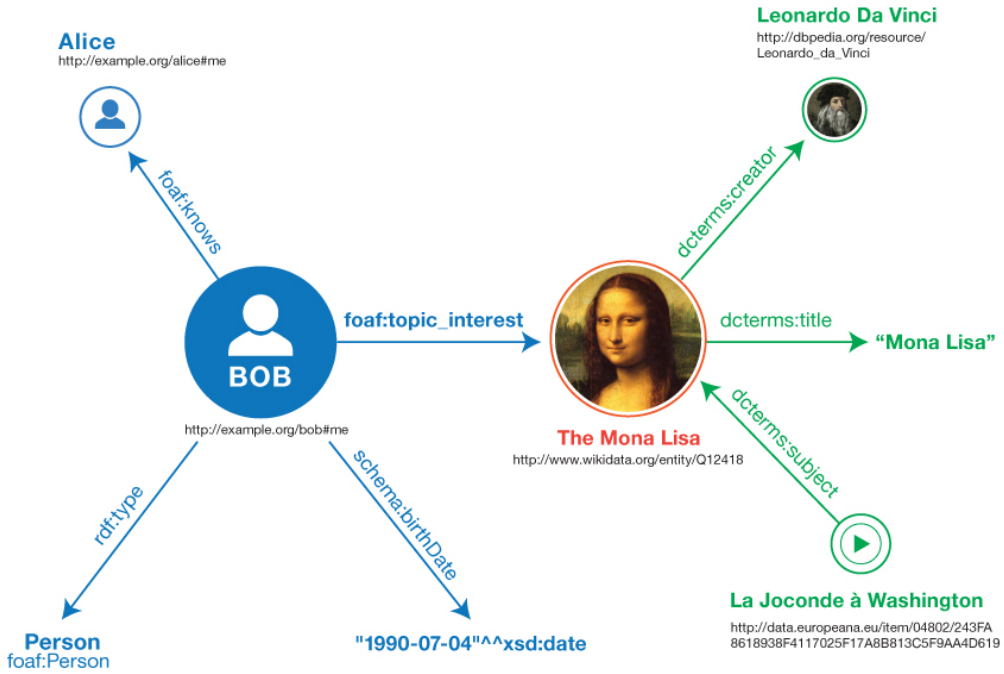
RDF, *RDFS* i pro *OWL*). Jazyky je tak možné škálovat a využít podle konkrétních požadavků na expresivitu znalostního modelu. [3] V další části se práce zaměří na konkrétní specifikace jednotlivých jazyků, které jsou pro tuto práci významné. Je důležité zejména získání metadat o získaných datech, kterými by mohl být datový typ získaného sloupce nebo případně i jeho popis.

2.2 Datový model RDF

Resource Description Framework *(RDF, systém popisu zdrojů)* slouží pro vyjádření informací o zdrojích v rámci sémantického webu [1]. Datový model *RDF* je založen na formátu trojic. Ukázka kódu 2.1 zobrazuje jednu trojici, která vyjadřuje orientovaný vztah mezi dvěma uzly grafu. Subjekt neboli zdroj *(subject)* a objekt *(object)* reprezentují dvě entity, mezi kterými je nějaký vztah. Povaha daného vztahu je vyjádřena právě predikátem *(predicate)* a udává vlastnost zdroje vůči objektu nebo jinému zdroji. RDF graf je pak množinou takovýchto trojic. [2]

```
1 <subject> <predicate> <object>
```

Zdrojový kód 2.1: Formát RDF trojic



Obrázek 2.2: Ukázka RDF grafu [2]

2.2.1 IRI

IRI (*Internationalized Resource Identifier*) je jednoznačným identifikátorem zdroje. V RDF trojici se může vyskytovat na libovolné pozici. Jednou z jeho podob je i *URL* (*Uniform Resource Locator*), který je používán pro adresy na webu s protokoly *HTTP* a *HTTPS*. Obecně *RDF* nepřiřazuje *IRI* význam, ten je získán až ze slovníků nebo konvencí, které danému *IRI* přísluší. *IRI* oproti *URI* umožňuje používat znaky libovolného jazyka. Je tedy navržen tak, aby byl schopen reprezentovat identifikátory zdrojů z různých jazyků. Toho dosahuje používáním *Unicode* kódování znaků. [1]

Příklad hodnoty *IRI* je *http://wikidata.org/entity/Q12418* na trojici z obrázku 2.2, jež představuje identifikátor obrazu *The Mona Lisa*. K tomuto zdroji přísluší predikát *creator* (*http://purl.org/dc/terms/creator*) s objektem *Leonardo Da Vinci* (*http://dbpedia.org/resource/Leonardo_da_Vinci*). Predikát určuje vlastnost tvůrce obrazu a objekt pak identifikátor daného tvůrce.

Znázornění plné formy *IRI* se dá zkrátit do prefixové notace, pokud je konkrétní prefix k danému IRI přiřazen. Příklad je znázorněn na obrázku 2.2 u hodnoty *dcterms:creator*, kdy plné znění by mělo podobu *http://purl.org/dc/terms/creator*. Zkratka *dcterms* tedy nahrazuje část *http://purl.org/dc/terms/*.

2.2.2 Literál

RDF literály jsou používány pro základní datové typy, vždy však na pozici objektu. Literál může být, ale nemusí, následován datovým typem. Datový typ je pak označen za pomocí *IRI*. Na konceptuální úrovni se jedná už o konkrétní atributy nebo vlastnosti zdroje. Všechny základní primitivní datové typy jako je *integer, boolean, dateTime* nebo *date* jsou poskytnuty ve slovníku *XML Schema* viz 2.4.1. Literály nejsou ale omezeny pouze na tyto základní datové typy. Obdobným způsobem lze odkazovat schéma datových typů z libovolného *IRI*, přichází v úvahu i definice vlastních datových typů. Za řetězcovým literálem se může nacházet i jazykový tag. [4]

```
1 "tao te ting"
2 "Ramana Maharshi"^^<http://www.w3.org/2001/XMLSchema#string>
3 "42"^^xsd:integer
4 'Beyond Enlightenment'@en
5 "xyz"^^ <https://ex.org/CustomDataType>
```

Zdrojový kód 2.2: Ukázka hodnot literálů

2.2.3 Anonymní uzel

Anonymní uzel v RDF je takový uzel, který nepopisuje žádný zdroj. Tvrzení, které v sobě má anonymní uzel, říká, že něco s takovým vztahem existuje, ale explicitně

to není pojmenováno. Takový stav může nastat jak v pozici subjektu, tak v pozici objektu. [1]

2.3 Formát RDF dat

Pro zapisování a čtení RDF dat existuje hned několik formátů, které budou následně popsány a k některým bude uveden i příklad. Pro kýženou aplikaci je potřeba zvolit vhodný formát, se kterým se bude pracovat a případně se do něj ostatní formáty budou převádět. Práce projde specifikace následujících formátů: *N-Triplets, Turtle, TriG, N-Quads, JSON-LD, RDF/XML*.

2.3.1 N-Triples

Základní formát, kdy každá trojice odpovídá trojici, která je zakončena tečkou a každý *uri* prvek (subjekt, predikát, objekt) je uzavřen znaky <>. Trojice, které byly doposud uvedeny, patřily k této formě. [5]

```
<http://ex.org/Frantisek>
<http://ex.org/livesIn> <http://ex.org/Assisi> .

<http://ex.org/Frantisek>
<http://xmlns.com/foaf/0.1/knows> <http://ex.org/Klara> .
```

Zdrojový kód 2.3: Ukázka formátu N-Triples

2.3.2 Turtle

Turtle navazuje na N-Triples, které rozšiřuje o možnost prefixů. IRI, u kterých není uveden prefix, jsou vyhodnoceny proti BASE prefixu. Množinu trojic, patřící k jednomu subjektu, lze zapsat v kratší formě, kdy je každá trojice oddělená středníkem a konec množiny posléze tečkou. [6]

```
BASE <http://ex.org>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

:Frantisek;
  :livesIn   :Assisi    ;
  foaf:knows   :Klara     .
```

Zdrojový kód 2.4: Ukázka formátu Turtle

2.3.3 TriG

Turtle notace, obdobně jako N-Triples, reprezentuje v jednom souboru pouze jeden graf. TriG rozšiřuje Turtle notaci o možnost reprezentování více pojmenovaných

grafů v rámci jednoho souboru. Za identifikátorem grafu následují ve složených závorkách jeho trojice. Na následující ukázce je uveden kromě základního grafu i pojmenovaný graf *:JeruzalemskyGraf*. [7]

```
BASE <http://ex.org>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX graphs <http://ex.org/graphs>

# default graph (AssisiGraf)
    {
      :Frantisek  :livesIn  :Assisi     .
      :Klara      :livesIn  :Assisi     .
    }

graphs:JeruzalemskyGraf
    {
       :Jezis     :livesIn  :Jeruzalem  .
    }
```

Zdrojový kód 2.5: Ukázka formátu TriG

2.3.4 N-Quads

N-Quads řeší problém s více grafy jiným způsobem než TriG. Místo trojic zde máme čtveřice, kdy po subjektu, predikátu a objektu, následuje název grafu. [8]

```
BASE <http://ex.org>
PREFIX g <http://ex.org/graphs>

<subjekt>  <predicate>  <object>    <graph>
:Jezis     :livesIn     :Jeruzalem   g:Graf1  .
```

Zdrojový kód 2.6: Ukázka formátu N-Quads

2.3.5 JSON-LD

JSON je formátem využívaným pro výměnu dat mezi webovými službami. JSON-LD (Javascript Object Notation - Linked Data) specifikuje syntax pro rozšíření JSON o propojená data. Základem JSON-LD dokumentu je specifikování kontextu (@context), ve kterém se nastaví prefixy používaných IRI. Následuje @id, kde je označení subjektu a dále jeho predikáty (JSON atributy) a objekty (JSON hodnoty atributů). V případě, že je potřeba popsat více subjektů v rámci jednoho JSON-LD, používá se @graph, který obsahuje pole objektů seskupených opět po jednom subjektu. [9]

```
{
  "@context": {
    "foaf": "http://xmlns.com/foaf/0.1/",
    "ex": "http://ex.org/"
  },
      "@id": "ex:Frantisek",
      "ex:livesIn": "ex:Assisi",
      "foaf:knows": "ex:Klara"
}
```

Zdrojový kód 2.7: Ukázka formátu JSON-LD

JSON-LD jako odpověď na SPARQL dotaz je serializován do formátu JSON jako jediný objekt na nejvyšší úrovni. Tento objekt má člen *head* a dále člen *results (SELECT)*, nebo člen *boolean (ASK)*, dle dotazu. Ukázková odpověď ASK dotazu v příloze E.1 pouze vrací hodnotu boolean značící, zda byl odpovídající graf nalezen, či nikoliv. [10]

U odpovědi SELECT dotazu v příloze E.2 lze vidět, že v *head* atributu odpovědi a parametru *vars* jsou vždy obsaženy názvy dotazovaných proměnných, odpovídající pořadí proměnných z dotazu. Následuje nepovinné pole *link*, kde mohou být obsaženy IRI, odkazující na více informací.

Objekt *results* obsahuje pole *bindings*, které zahrnuje záznamy, což jsou objekty s atributy pojmenovanými podle názvů dotazovaných proměnných. Každé dotazované proměnné pro daný záznam přísluší hodnota *type*, která udává, zda se jedná o *literál* či *uri/iri*. Kromě těchto dvou případů může nastat i hodnota *anonymního uzlu*, která by měla hodnotu *bnode*. Pokud se jedná o literál, může být uveden i jeho datový typ. Pokud zde není, předpokládá se, že se jedná automaticky o *string* jako u prvního objektu. String může být rozšířen o jazykový tag *xml:lang*. [10]

2.3.6 RDF/XML

RDF/XML vychází z *XML (extensible markup language)* syntaxe. Každý element uvnitř *rdf:Description* pak odpovídá jedné trojici. [11]

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns="http://ex.org/">
  <rdf:Description rdf:about="http://ex.org/Frantisek">
    <livesIn rdf:resource="http://ex.org/Assisi"/>
    <foaf:knows rdf:resource="http://ex.org/Klara"/>
  </rdf:Description>
</rdf:RDF>
```

Zdrojový kód 2.8: Ukázka formátu RDF/XML

2.4 Ontologie a slovníky

2.4.1 RDF slovníky

Jak bylo ukázáno v kapitole 2.1, RDF sám o sobě poskytuje pouze strukturovaný formát pro data, ale nenabízí žádné informace o sémantice těchto dat. Pro přidání významu je zapotřebí použít zdroje, které informace o významu konceptů a vztahů poskytují. Takové zdroje jsou známé jako RDF slovníky. Popisují klíčové koncepty a jejich vztahy.[1]

Tyto kolekce IRI jsou znovupoužitelné pro definování zájmové oblasti v jiném slovníku. Dochází k charakterizaci, klasifikaci a limitaci použití jednotlivých konceptů a vztahů. Je ve zvyku, že ve slovnících IRI začínají stejným podřetězcem, který se nazývá jmenným prostorem (*IRI namespace)*. Tento jmenný prostor pak nese svůj prefix. Jednoduchým příkladem využití je reference na primitivní datové typy ze slovníku *XSD* (viz 2.2.2). Tabulka 2.1 ukazuje několik základních slovníků (z *W3 Common Vocabularies*). [1]

Tabulka 2.1: Přehled základních RDF slovníků

Prefix	IRI	Slovník
xsd	<http://www.w3.org/2001/XMLSchema#>	XML Schema
rdf	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>	RDF Vocabulary
rdfs	<http://www.w3.org/2000/01/rdf-schema#>	RDF Schema
owl	<http://www.w3.org/2002/07/owl#>	OWL
foaf	<http://xmlns.com/foaf/0.1/>	FOAF
dc	<http://purl.org/dc/elements/1.1/>	Dublin Core

2.4.2 RDF Schema

Resource Description Framework Schema (RDFS) je slovníkem, který definuje konstrukty pro vyjádření tříd objektů, jejich vlastností a hierarchií. Zavádí klíčové pojmy *class* (třída) a *property* (vlastnost). Třída v RDFS představuje skupinu objektů se společnými vlastnostmi a objekty této třídy se nazývají jejími instancemi. [4]

Tabulka 2.2: Přehled základních RDFS tříd

Název třídy	Popis
rdfs:Resource	Třída zdrojů
rdfs:Literal	Třída literálů
rdfs:Class	Třída tříd
rdfs:Property	Třída RDF vlastností
rdfs:Datatype	Třída RDF datových typů

Tabulka 2.3: Přehled základních RDFS vlastností

Název vlastnosti	Popis
rdfs:subclassOf	Modelování hierarchie mezi třídami
rdfs:subPropertyOf	Modelování hierarchie mezi vlastnostmi
rdfs:domain	Stanovuje třídu subjektu
rdfs:range	Stanovuje třídu objektu
rdfs:label	Udává vlastnost lidsky přívětivého popisu atributu

Ukázka definování třídy *Enlightenment* pomocí RDFS konstrukcí:

```
BASE <http://ex.org/classes>

:Enlightenment rdf:type rdfs:Class.
# je podtřídou StateOfBeing
:Enlightenment rdfs:subClassOf :StateOfBeing.
# popis v češtině
:Enlightenment rdfs:label "Osvícení"@cs.

:hasCompassion rdf:type rdf:Property
# tuto vlastnost má třída Enlightenment
:hasCompassion rdfs:domain :Enlightenment.
# má jí k jakékoliv instanci Resource
:hasCompassion rdfs:range rdfs:Resource.
```

Zdrojový kód 2.9: Definování třídy pomocí RDFS

2.4.3 OWL

Web Ontology Language (OWL) již definuje pravidla na úrovni deskripční logiky. Jazyk je navržen pro reprezentaci bohatších a komplexnějších vědomostí o věcech,

skupinách věcí a relacemi mezi nimi. [12] Samotný termín ontologie pochází z řeckých slov *"ontos"* (bytí) a *"logos"* (učení). V informatice je Gruberem ontologie definována jako: *"formální specifikace sdílené konceptualizace"*. [13]

Oproti RDFS přidává OWL relace mezi třídami, kardinalitu, kvantifikátory, rovnost, disjunktnost, ekvivalenci tříd, více typů vlastností, charakteristiky a další. Hlavním stavebním kamenem OWL jsou axiomy (základní výroky), entity (prvky odkazující na objekty reálného světa) a výrazy (kombinace axiomů a entit). Vyjádření vědomostí v rámci OWL se označuje jako trvrzení (statement). Příkladem může být tvrzení: *"Každý člověk je smrtelný"*. Každá vědomost je potom tvořena kolekcí souvisejících tvrzení a každé tvrzení je následkem jiného tvrzení. Tvrzení je tedy pravdivé, pokud ostatní tvrzení jsou pravdivá. [12]

V rámci OWL jsou rozlišovány komponenty jedince *(individual, představuje objekt zájmu)*, vlastnosti *(property, představuje vztah mezi jedinci)* a třídy *(class, představuje množinu jedinců se stejnými vlastnostmi)*. Jak bylo nastíněno v kapitole 2.1, existuje více úrovní jazyka OWL, kdy každá hierarchicky vyšší verze přidává komplexnější konstrukce. Jedná se o úrovně OWL, OWL DL a OWL FULL.

SPARQL a SPARQL Endpoint — 3

SPARQL *(SPARQL Protocol and RDF Query Language)* je dotazovací jazyk a protokol pro sémantické dotazování a manipulaci s RDF daty. Jedná se o standardizovaný jazyk pro práci s RDF. SPARQL definuje dotazy *SELECT, DESCRIBE, ASK, CONSTRUCT* a od verze SPARQL 1.1 umožňuje i komplikovanější konstrukce (např. agregace) a aktualizační dotazy *DELETE, INSERT, LOAD, CLEAR*. [14] Práce se zabývá výběrovými dotazy. V následující části bude popsána obecná struktura SPARQL dotazu a poté SPARQL Endpoint a jeho možnosti.

3.1 Struktura SPARQL dotazu

Syntaxe SPARQL dotazu připomíná jazyk SQL. Poskytnutý vzor grafu v klauzuli WHERE odpovídá formátu *Turtle* (viz 2.3.2). SPARQL dotaz má následující strukturu (komentáře se označují znakem #), která bude po částech podrobněji vysvětlena:

```
1  PREFIX ex: <http://ex.org/> # prefix
2  SELECT ?athleteID ?predicate ?object # typ dotazu
3  FROM NAMED <http://ex.org/graph> # zdroj dat
4  WHERE # grafový vzor
5  {
6      ?athlete rdf:type ex:Athlete .
7      ?athlete ex:hasAge ?age.
8      FILTER (?age > 18)
9  }
10 GROUP BY ?athlete HAVING (SUM(?age) < 50) # agregace
11 LIMIT 50 # modifikator
```

Zdrojový kód 3.1: Struktura SPARQL dotazu

Volitelná část dotazu *prefix* specifikuje zkratky *IRI* identifikátorů, které pak mohou být v rámci dotazu použity pro zkrácenou formu zápisu plného IRI (viz 2.2.1). Následuje povinná část, která určuje *typ dotazu* a jeho potřebné parametry. Část *zdroj dat* je volitelnou specifikací, z jakých zdrojů dat má SPARQL dotaz čerpat. Může zahrnovat jména grafů nebo datových sad. Povinná část *grafový vzor* je obsažena

v klauzuli *WHERE*. Definuje hledaný vzor grafu, kterému musí data odpovídat. *Agregace* je pak volitelná část dotazu, která umožňuje provádět agregační funkce jako *SUM (suma), COUNT (počet), AVG (průměr)* a další. Poslední volitelná část dotazu je *modifikátor*, který určuje, jak bude počet výsledků omezen nebo jaké bude pořadí výsledků. Jedná se o *LIMIT (maximální počet výsledků), OFFSET (přeskočení výsledků), ORDER BY (řazení výsledků), DISTINCT (zajištění unikátních hodnot)* a další.

3.2 SPARQL Endpoint

SPARQL Endpoint je přístupový bod, který umožňuje vzdálené dotazování a správu dat z RDF úložišť pomocí SPARQL dotazů. Existují různé varianty implementací RDF úložišť, jako jsou OpenLink Virtuoso [15], Apache Jena Fuseki [16] a další. Tyto služby mohou podporovat různé modely uchování dat, včetně persistentních a nepersistentních (in-memory) variant. [14]

Endpoint je dosažitelný pomocí *HTTP* nebo *HTTPS* protokolu a může být i privátní a omezený autorizačními metodami. Endpoint naslouchá a zpracovává příchozí SPARQL dotaz jako webové API. Existuje mnoho dostupných endpointů, které organizace veřejně poskytují. [14] Příklady SPARQL Endpointů:

Tabulka 3.1: Příklad SPARQL Endpointů

Název	**Adresa**
DBpedia	http://dbpedia.org/sparql
Wikidata	https://query.wikidata.org/sparql
RUIAN	https://ruian.linked.opendata.cz/sparql
Česká národní Open Data	https://data.gov.cz/sparql
Ministerstvo financí ČR	https://opendata.mfcr.cz/pages/sparql

SPARQL Endpoint se může lišit v podpoře SPARQL 1.1 a nebo i pokročilejšího SPARQL 1.1 [14]. SPARQL Endpoint umožňuje získat odpovědi na dotazy dle implementace v některých z formátů, které byly zmíněny v kapitole 2.3. Konkrétně byla blíže prozkoumána odpověď SPARQL Endpointu ve formátu *JSON-LD* (viz 2.3.5). Popsaná metadata budou pak obsažena v jiné serializaci i u ostatních formátů.

Metody analýzy a vizualizace dat

4

Analýza dat je definována jako proces čištění, transformace a modelování dat za účelem zjištění užitečných informací, na jejichž základě je možné přijmout rozhodnutí. Výsledky analýzy jsou představeny ve srozumitelných reportech, vizualizacích a přehledech. [17]

Rozdělení metod analýzy a vizualizace dat v rámci práce odpovídá základním procesům práce s daty, které jsou obvykle implementovány v aplikacích, jež se analýzou datových sad zabývají. Jednotlivé procesy budou následně blíže popsány.

- *Získání dat (Extraction)*
- *Transformace dat (Transformation)*
- *Vizualizace dat*
- *Interakce s vizuály*
- *Dashboardy*
- *Statistická analýza*

4.1 Získání dat

První fází každé práce s daty je jejich získání neboli extrakce dat ze zdroje. Zdrojů dat existuje mnoho a jsou poskytovány v rozdílných formátech, které jsou poskytovány různými typy úložišť. Cílem extrakce je tedy správně přečíst zdrojová data podle předem připraveného schématu, a umožnit přechod do další částí procesu. Aplikace musí zvládat specifikované datové typy správně přečíst, aby se s nimi v dalších fázích mohlo pracovat. Typy zdrojů: [18]

- *Soubory:* Textové soubory (csv, txt), excelové soubory (xls, xlsx), JSON a další.
- *Databáze:* Relační databáze (MySQL, PostgreSQL, Oracle), noSQL databáze (MongoDB, Cassandra), datové sklady a další.

- *Web:* Webové služby (API, SPARQL Endpoint), web scraping a další.
- *Cloud:* Google Cloud Storage, Amazon S3 a další.
- *Další zdroje dat:* IoT zařízení, geografická data, data ze sociálních sítí, biometrická data a další.

4.2 Transformace dat

Data mohou nabývat různých formátů v různé kvalitě. Cílem transformace je připravit data do podoby, ve které s nimi bude možné pracovat. Kvalitní transformací dojde k zajištění úplnosti, správnosti a vhodnosti dat pro další potřeby, nebo by se naopak mělo odhalit, že jsou data nepoužitelná. Základní operace transformace dat dle [18]:

- *Validace:* Ověření integrity dat (null hodnoty, rozsah hodnot, datové typy).
- *Odstranění duplicit:* Zamezení redundance v datovém souboru.
- *Čištění dat:* Zpracování chybných hodnot a neplatných dat.
- *Filtrování dat:* Zaměření se na podmnožinu dat nebo redukování objemu dat.
- *Konverze datových typů, vzdáleností a jednotek:* Převedení hodnot do požadovaného formátu.
- *Normalizace dat:* Snížení redundance dat a zajištění jejich konzistence, každý fakt je uložen pouze na jednom místě.
- *Denormalizace dat:* Opak normalizace, záměrné přidání odvozených hodnot.
- *Shlukování dat:* Spojení dat z různých zdrojů do jedné tabulky.
- *Odvození nových hodnot:* Vytvoření nových hodnot na základě existujících dat.
- *Agregace:* Seskupení záznamů a získání jejich souhrnných informací.

4.3 Vizualizace dat

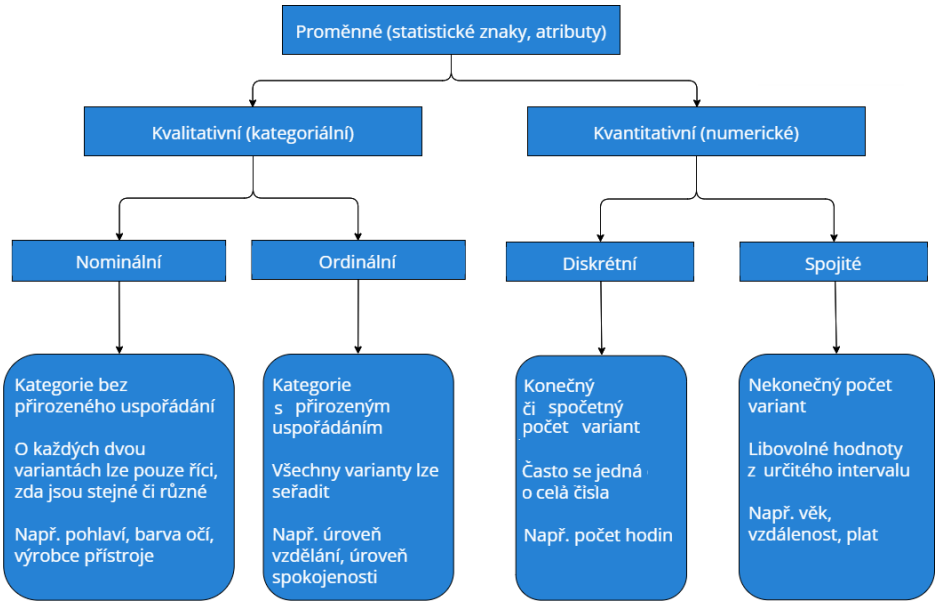
Připravená data je možné graficky zobrazit v podobě tabulky nebo grafu. Vizualizace dat je dle [19] definována jako *„používání počítačem podporovaných, interaktivních, vizuálních reprezentací dat za účelem zesílení poznávání“*. Obecná definice zdůrazňuje vizualizaci jako práci s daty s cílem většího porozumění datům. Fáze vizualizace a transformace se může i prolínat.

4.3.1 Parametry volby vizuálu

Volba správného vizuálu je klíčovým krokem při prezentaci dat, neboť může výrazně ovlivnit efektivitu sdělení a porozumění informacím. Při rozhodování o vhodném vizuálním formátu je důležité zvážit povahu dat a cíle prezentace.

Datové typy proměnných

Stěžejním je identifikace datového typu sloupců, které je potřeba zobrazit. Proměnné mohou být kvalitativní, tedy nominální (např. typ květiny) nebo ordinální (např. úroveň vzdělání), a kvantitativní (např. délka vlasu či hmotnost slona) [20]. Rozdělení dle [20] a příklad hodnot je znázorněn na obrázku 4.1.



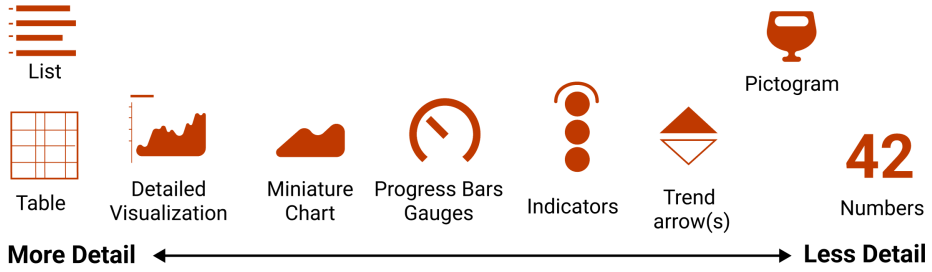
Obrázek 4.1: Datové typy proměnných

Kategoriální proměnné budou v případě grafu sloužit jako popisky nebo třídy, podle kterých se budou data shlukovat. Numerické hodnoty mohou poskytovat hodnoty daných skupin. Například časové trendy mohou být nejlépe zachyceny čárovým grafem, zatímco porovnání jednotlivých kategorií z celku může vyžadovat výsečový nebo prstencový graf. Pro porovnání hodnot mezi dvěma číselnými proměnnými může být efektivní bodový nebo bublinový graf.

Míra podrobnosti

Dalším faktorem je komplexita informací a míra podrobnosti, která má být vizuálem reprezentována. Pro rychlé porozumění trendu může být vhodný jednoduchý sloupcový nebo plošný graf. Naopak, pokud je potřeba detailní analýza, mohou

být preferovány podrobnější grafy nebo i tabulky. [21] Obrázek 4.2 toto rozdělení znázorňuje:



Obrázek 4.2: Vizuály dle míry podrobnosti z Dashboard Design Patterns [21]

Více informací může být zaneseno do grafu použitím rozdílných barev, kdy například rozdílné třídy záznamů mají v grafu jiné barvy, jak je znázorněno na obrázku 4.3:

Obrázek 4.3: Vizuály dle barvy z Dashboard Design Patterns [21]

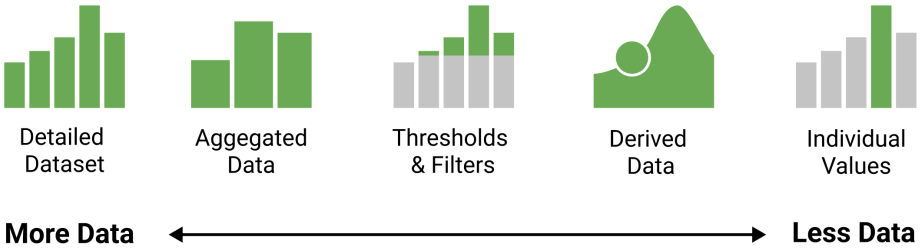
Velikost datového souboru

Volba vizuálu, či nejméně jeho parametrů a detailnosti, se bude lišit i v případě rozdílné velikosti datového souboru. Jiným způsobem bude prezentován pohled na všechna data, momentální filtrovaný pohled nebo pohled na jediný záznam, jak ukazuje obrázek 4.4 [21]:

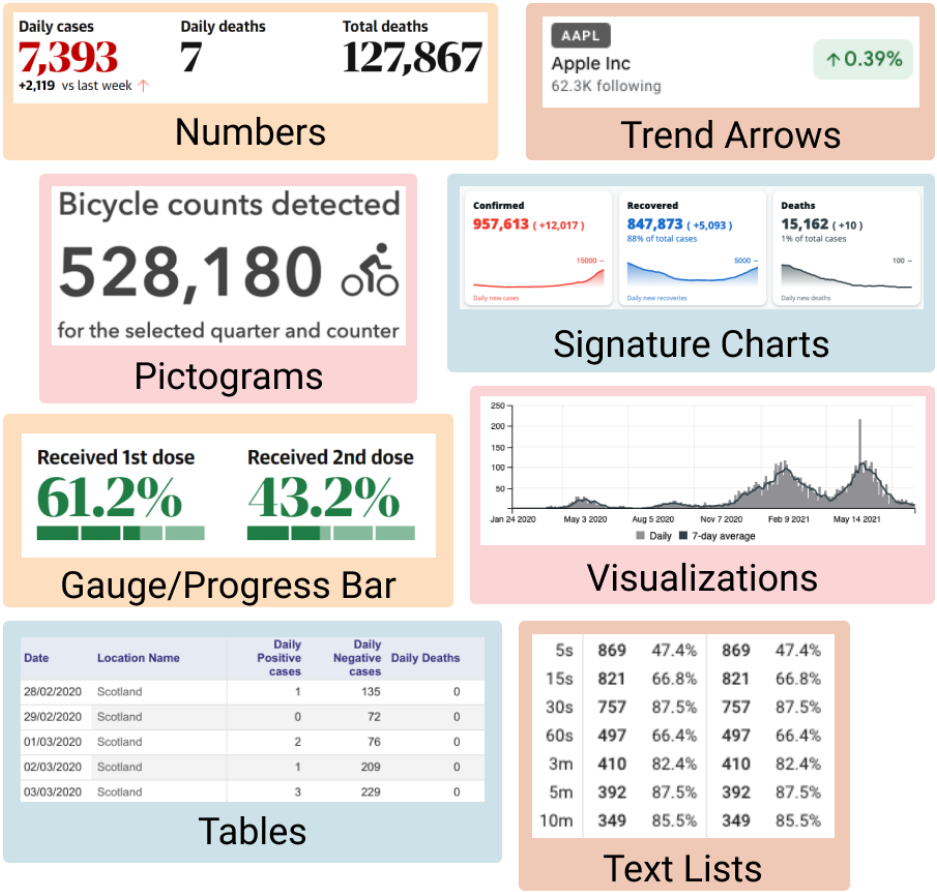
4.3.2 Typy vizuálů

Typy vizuálů viz obrázek 4.5 [21]:

- *Čísla (numbers):* Vizuál může být jednoduchý, jako zobrazení jednoho klíčového čísla (např. procentuální zastoupení nemocných).
- *Šipky trendu (trend arrow(s)):* Vizuál reprezentující indikaci změny hodnoty v datech (např. růst inflace udaný směrem a sytostí barvy šipky).



Obrázek 4.4: Vizuály dle množství dat z Dashboard Design Patterns [21]



Obrázek 4.5: Typy vizuálů z Dashboard Design Patterns [21]

- *Piktogram (pictogram):* Symbol ilustrující koncept v datech (např. příslušnost k dané skupině).
- *Měřítka a pruh postupu (progress bars gauges, indicators):* Vizuál používaný k reprezentaci škály z rozsahu hodnot (např. nákaza na 100 obyvatel).

- *Přehledový graf (signature charts):* Malé stručné grafy bez popisů os, štítků nebo značek cílem je rychlé porozumění trendu na základě vizualizace.
- *Podrobný graf (detailed visualization):* Graf s dostatečnými podrobnostmi (osy, značky, hodnoty) pro bližší porozumění trendu i konkrétním hodnotám.
- *Tabulka (table):* Zobrazení dat v surovém tabulkovém formátu.
- *Textové seznamy (text lists):* Seznam textových informací zobrazující kvantitativní informace.

Jednotlivé vizuály se často navzájem kombinují, kdy například v buňce tabulky může být obsažena šipka trendu nebo i malý přehledový graf.

4.3.3 Typy grafů

Graf je základním schématem pro vizualizaci dat a jedná se o ověřený způsob grafické reprezentace kvantitativních dat. Jelikož se jedná o komplikovanější vizuál než ostatní, které byly uvedeny, budou blíže rozebrány základní typy grafů: [21]

- *Sloupcový graf:* Srovnání hodnot nezávislých proměnných.
- *Spojnicový graf:* Kontinuální data, horizontální osa obvykle představuje časové období, graf ukazuje, jak se proměnná vyvíjí v průběhu času.
- *Plošný graf:* Obdoba spojnicového grafu, ale vykreslená oblast je součtem všech proměnných pocházejících z různých oblastí.
- *Výsečový graf:* Rychlé srovnání poměrových dat, použití v případě, kdy data představují části celku.
- *Prstencový graf:* Podobně jako výsečový, ale může porovnávat více než jednu sadu dat.
- *Bodový graf:* Zobrazuje hodnoty dvou proměnných, často používaný pro nalezení vztahu mezi daty.
- *Bublinový graf:* Rozšiřuje bodový graf o třetí proměnnou zobrazenou velikostí samotného bodu (bubliny).

4.4 Interakce s vizuály

Je to možnost interaktivně s vizuály pracovat a měnit jejich parametry, která dodává pravou hodnotu datové analýze s využitím vizuálů. Analytik vybere základní parametry a získá vizuální reprezentaci dat. Na jejím základě zavrhne či přijme představy o datech a může pokračovat znovu s upraveným parametry, jiným vizuálem nebo jinými předpoklady. Průzkum dat vizuálními metodami je procesem experimentování a objevování pomocí živé interakce s těmito vizuály. Proces vizualizační explorace může být rozdělen následujícím způsobem [21]:

- *Overview first:* Široký pohled na data, uchopení celkového kontextu dat.
- *Zoom and filter:* Přiblížení se konkrétním částem dat, lepší pochopení podrobností a vztahů mezi daty pomocí filtrů.
- *Details on demand:* Podrobnější informace o konkrétní části dat získatelné jednoduchým dotazem nebo kliknutím.

Základními možnostmi interakce jsou: filtrování, přiblížení a dynamické řazení. V případě, že je přítomno více vizuálů, nastává možnost propojení filtrů na celou skupinu vizuálů, kdy například přiblížení nebo filtrace ovlivní hned několik vizuálů. Interakce je možná přímo v nastavení vizuálů nebo použitím samostatného ovládacího prvku pro danou funkcionalitu. Příkladem takového ovládacího prvku může být posuvník nebo rozbalovací seznam.

4.5 Dashboard

Dashboardem lze označit uspořádání několika souvisejících vizuálů. Jedná se o grafické zobrazení nejdůležitějších informací potřebných k dosažení jednoho nebo více cílů, které jsou uspořádány na jediné obrazovce, aby se informace daly monitorovat hned na první pohled. Prostoru na dashboardu není příliš a je důležité zvolit jen vhodné vizuály, které spolu dohromady tvoří relevantní informaci. [21]

4.5.1 Rozložení dashboardu

U dashboardů se lze setkat s různými druhy rozložení jejich vizuálů, jak je znázorněno na obrázku 4.6:

- *Otevřené rozložení (Open):* Jednotlivé vizuály jsou uspořádány bez zjevných pravidel. Často jsou zarovnány do mřížek, ale neexistuje rozlišení významu sousednosti vizuálů. Mají rovnocenný význam. [21]

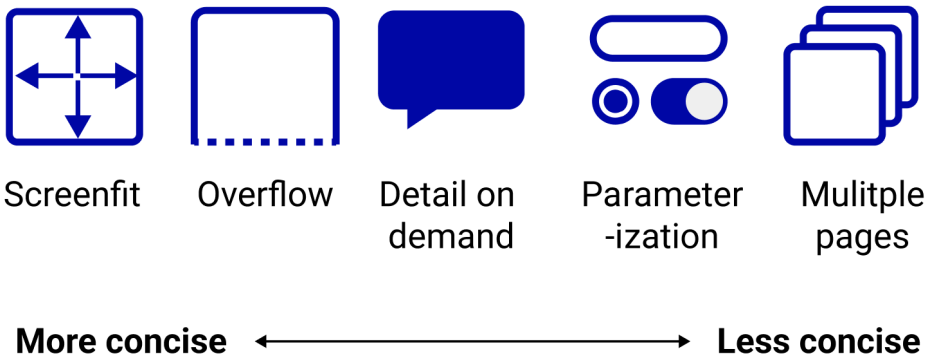
Obrázek 4.6: Typy rozložení dashboardu z Dashboard Design Patterns [21]

- *Tabulkové rozložení (Table):* Zarovnává vizuály do sloupců a řádků, kdy každý má svůj konkrétní význam a vyvolává opakování informací. Řádky a sloupce mohou reprezentovat aspekty datové sady jako je čas, datové prvky nebo příslušnost k nějaké skupině. [21]
- *Vrstvené rozložení (Stratified):* Zdůrazňuje postupné uspořádání vizuálů a jejich informací shora dolů. Obecná informace může být zobrazena nahoře, zatímco pod ní budou zobrazeny podrobnější informace. [21]
- *Seskupená rozložení (Grouped):* Seskupují dva nebo více vizuálů s konkrétním vztahem. Může nastat například použitím stejného pozadí pro zdůraznění seskupení nebo použití ohraničení. [21]
- *Schématická rozložení (Schematic):* Umísťují vizuály na dashboard například dle fyzického uspořádání nebo jiného typu schématických vztahů. Může se jednat i o vizuály umístěné na mapě k reprezentaci geografické polohy, využívající uživatelovu znalost prostoru k podpoře vizuální informace. [21]

4.5.2 Prostor dashboardu na obrazovce

Dalším faktorem dashboardu je prostor, který je pro něj dostupný na obrazovce. Na dashboard se vejde jen omezené množství vizuálů a tedy i informací. Zatímco některé dashboardy se vejdou na obrazovku celé, pro některé je potřeba použít stručnější způsob. Způsoby využití prostoru obrazovky dashboardem jsou znázorněny na obrázku 4.7 [21]:

- *Přizpůsobení obrazovce (Screenfit):* Plně se vejde na obrazovku a má všechny své informace kdykoliv viditelné. K prohlížení obsahu není potřeba interakce s dashboardem (posouvání, změna stránek). Veškerá informace je přítomná. [21]
- *Přetečení (Overflow):* Použití umožňuje dashboardu překročit prostor obrazovky tím, že je obsah vykreslen mimo obrazovku. Tyto informace jsou zobrazeny až posouváním stránky uživatelem. [21]



Obrázek 4.7: Prostor dashboardu z Dashboard Design Patterns [21]

- *Podrobnosti na vyžádání (Detail on demand):* Pro snížení množství informací na obrazovce se používá přístup, kdy jsou dodatečné informace zobrazeny pouze vyžádáním uživatele, například kliknutím (tooltip, vyskakovací okno). [21]
- *Parametrizace (Parametrization):* Přináší s sebou ovládací prvky, kde se pomocí rozbalovacích menu, posuvníků a dalších vybere konkrétní podmnožina zobrazených dat, která je v dashboardu obsažena. [21]
- *Více stránek (Multiple pages):* Přístup použití několika stránek zobrazuje vždy jen jednu stránku najednou. Jedna stránka pak odpovídá zpravidla jednomu kompozitnímu pohledu uvnitř dashboardu. Existuje pak nějaký způsob navigace mezi těmito stránkami (panel, navigační tlačítka, odkazy). [21]

4.5.3 Sdílení dashboardu

Úkolem dashboardu je přehledně prezentovat určité informace a jeho hlavním cílem je umožnit publikaci a sdílení těchto informací. Praxe výrazně usnadňuje efektivní prezentaci dat a analýz ve srozumitelné podobě. V případě, že je dashboard kvalitně navržen a implementován, může sloužit i uživatelům, kteří nemají hlubší znalost daného tématu. Umožňuje jim snadno získávat potřebné informace.

Mnoho vizualizačních aplikací poskytuje vedle tvorby dashboardů také možnosti sdílení obsahu mimo rámec aplikace. Jedná se například o vnoření dashboardu do vlastní webové stránky pomocí HTML iframe, což umožňuje širšímu publiku snadný přístup k prezentovaným datům. Možnost sdílení dashboardu zvyšuje dostupnost informací a otevírá prostor efektivnější komunikaci a spolupráci ve sdíleném prostoru.

4.6 Statistická analýza

Statistická analýza hraje klíčovou roli v interpretaci a porozumění datových sad. Lze říci, že zmíněné vizualizace a transformace jsou dílčí součástí statistické analýzy nebo přípravy dat pro tuto analýzu. Pro úplnost je ale nutné zmínit i další prostředky, které statistická analýza může zahrnovat. Základní kategorie statistické analýzy [22]:

- *Deskriptivní analýza (popisná):* Popisná statistika (průměry, mediány, rozptyly, kvartily, histogramy, krabicové grafy), grafická prezentace dat (vizualizace distribuce, centrální tendence a variability).
- *Inferenční analýza (vyhledávání zákonitostí):* Statistické testy (T-testy, ANOVA, chi-kvadrát testy), konfidenční intervaly (odhad pravděpodobného rozmezí hodnot pro daný parametr), regresní analýza (identifikace a kvantifikace vztahu mezi proměnnými).
- *Další techniky:* Regresní modely (lineární regrese, nelineární regrese), analýza časových řad (identifikace trendů, sezónních vzorů, autoregresní modely), analýza faktorů (faktorová analýza pro odhalení skrytých faktorů), klasifikační modely (rozhodovací stromy, k-nearest neighbors).

Nástroje analýzy a vizualizace dat — 5

Nástroje analýzy a vizualizace dat pro zkoumání byly vybrány po konzultaci s vedoucím práce a na základě průzkumu nejznámějších aplikací zabývajících se touto problematikou. V rámci zkoumání existujících nástrojů, řešících danou problematiku, byly prozkoumány i další nástroje (např. *Grafana*[1], *Tibco Statistica*[2] nebo *YasGUI*)[3]. Pro hlubší průzkum analytické části byly zvoleny aplikace, které se dostatečně odlišují stylem, jakým problematiku řeší. Je možnost inspirovat se ve více ohledech a využít znalosti silných stránek každé z aplikací.

Cílem je zjištění podpory dat ze SPARQL Endpointu jednotlivých aplikací a ověření způsobu, jakým aplikace transformují svá data pro použití v dashboardech, a jak následná interakce s dashboardy a vizuály probíhá. Následně budou prozkoumány aplikace *Power BI*[4], *Looker Studio*[5] a *Tableau*[6]. Budou prozkoumány knihovny, které mohou být použity pro vývoj webové aplikace interaktivní vizualizace dat.

5.1 Popis aplikací

Power BI je produktem firmy *Microsoft* a už z názvu je jasné, že se jedná o software zabývající se *business intelligence*. Jedná se o kolekci softwarových služeb, aplikací a konektorů, které spolupracují, aby přeměnily nesouvisející zdroje dat na koherentní, vizuálně zajímavé interaktivní přehledy. Data můžou být ve formě excelové tabulky nebo kolekce cloudových a místních hybridních datových skladů. Power BI umožňuje připojit se ke zdrojům dat, vizualizovat je a zjistit, co je důležité, případně vizualizace veřejně sdílet. Konkrétní možnosti Power BI budou zkoumány na Microsoft Power BI Desktop 2.123.742.0 64-bit *(listopad 2023)*, který slouží právě

[1] Grafana - https://grafana.com/
[2] Tibco Statistica - https://statistica.pro/produkty-category/tibco-desktop
[3] YasGUI - https://yasgui.org/
[4] Power BI - https://www.microsoft.com/cs-cz/power-platform/products/power-bi
[5] Looker Studio - https://lookerstudio.google.com/
[6] Tableau - https://www.tableau.com/

k transformaci dat, vytváření datových modelů, vizualizací a sestav. Publikace dashboardů poskytuje webová aplikace Power BI. [23]

Looker Studio (dříve Google Data Studio) funguje jako efektivní online nástroj pro převod dat do vizuálních informativních sestav a dashboardů. Původně vzniklo v roce 2016 jako Google Data Studio, ale v říjnu 2022 Google tento nástroj přejmenoval na Looker Studio. Produkty fungovaly zpočátku samostatně. Google Data Studio bylo levnou možností při propojení zdrojů dat a vytváření řídících panelů. Narozdíl od Power BI běží Looker Studio zcela jako webová aplikace. Vytváření sestav a správu jejich datových zdrojů tak uživatel provádí prostřednictvím prohlížeče a nemusí si nic stahovat. Looker Studio je kategorizováno jako nástroj *web analytics* a *data vizualizations*. [24]

Společnost *Tableau* byla založena v roce 2003. Jejím cílem bylo zlepšení průběhu analýzy a zpřístupnění dat lidem pomocí vizualizace. Spoluzakladatelé Chris Stolte, Pat Hanrahan a Christian Chabot vyvinuli a patentovali základní technologii společnosti Tableau, VizQL, která vizuálně vyjadřuje data tím, že převádí akce přetahování na datové dotazy prostřednictvím intuitivního rozhraní. Aplikace zahrnuje strojové učení, statistiku, přirozený jazyk a inteligentní přípravu dat, které jsou užitečnější pro rozšíření lidské kreativity při analýze. Aplikace dle tvůrců nenabízí pouze kompletní integrovanou analytickou platformu, ale také osvědčené zdroje pro podporu, tím zákazníkům pomáhají zavádět a rozšiřovat kulturu založenou na datech a zvyšovat jejich odolnost a hodnotu díky silným výsledkům. Společnost Tableau byla v roce 2019 převzata společností Salesforce. Tableau lze rozdělit na webové služby Tableau Cloud (nově vzniklá webová aplikace pro datové analýzy a vizualizace), Tableau Server (správa přístupu k vytvářeným datovým zdrojům a dashboardům, publikace vizualizací) a dále Tableau Desktop (dlouhodobě vyvíjená desktopová aplikace, kategorizovaná jako *business intelligence*, *data analytics* a *data visualizations*). [25] Zpracování dat a vytváření dashboardů bylo testováno právě v aplikaci Tableau Desktop 2024.1 (*leden 2024*). [26]

5.2 Porovnání cen

Základní služby *Power BI* jsou poskytovány zdarma, s podmínkou nutné registrace. Přístup k modelování a interaktivnímu průzkumu dat je tedy pomocí *Power BI Desktop* možný i v bezplatné verzi. V případě, že potřebuje uživatel publikování sestav nebo řídících panelů a prohlížení daného obsahu v rámci celé organizace, je nutné zakoupit verzi Power BI Pro za 9,40€ měsíčně na uživatele *(k 8. 12. 2023)*. Pokud nastane i potřeba komplikovanějších modelů a častějších operací čtení a zápisu, lze zakoupit verzi Power BI Premium za 18,70€ měsíčně na uživatele *(k 8. 12. 2023)*. [27]

Looker Studio je *(k 29.12.2023)* zdarma, jen využití některých pokročilých vizuálů, které jsou vyvíjeny komunitou, je placené.

Tableau je placené a práce v něm byla testována v rámci zkušebního období. Pro používání nabízí tři základní cenové kategorie, které se liší v možnostech využití Tableau nástrojů pro práci s daty, sdílení těchto dat apod. Ceny *(k 30.4.2024)*: Tableau Viewer - 15$ měsíčně (zobrazovaní a interakce s dashboardy), Tableau Explorer - 42$ měsíčně (interakce s dashboardy a úprava dashboardů), Tableau Creator - 75$ měsíčně (veškerá funkcionalita včetně správy datových toků atd.). [28]

5.3 Porovnání podpory SPARQL Endpointu

5.3.1 Zpracování RDF formátů

RDF: Přímé zpracování souborů ve formátu N-Triplets, Turtle, TriG, N-Quads a RDFa žádná z aplikací nepodporuje.

RDF/XML: Ačkoliv v aplikaci Power BI, Looker Studio i Tableau existují možnosti pro čtení formátu XML, nebyl v žádné z aplikací nalezen přímočarý způsob, jak převést na tabulková data i RDF/XML.

JSON-LD: *Tableau i Power BI* podporují čtení souborů ve formátu JSON. Pro JSON-LD však přímá podpora neexistuje. V rámci transformací ale existují relativně přímočaré kroky, jak transformace na použitelnou tabulku z JSON-LD docílit. Zdroj se přečte jako JSON a dále je nutné převést atribut obsahující výsledky na tabulku. Následně lze z tabulky vybrat atributy obsahující hodnoty odpovídající datům z odpovědi na SPARQL dotaz. U obou aplikací je nevýhodou nutnost rozdílných kroků transformace pro JSON-LD s jinými atributy. Zde je pak nutné vytvořit novou transformaci, v případě Tableau novou konfiguraci čtení zdrojového souboru. V obou případech se výsledek transformace jeví velice použitelně, protože při správném rozbalení může být v dalších sloupcích zachována i úplná informace o hodnotě sloupce, včetně typu (uri/literál), datového typu a případně jazykového tagu.

Konektory pro JSON v *Looker Studio* existují. Přívětivý způsob, jak JSON-LD získaný ze SPARQL Endpointu převést do podoby, se kterou se dá v aplikaci následně pracovat, nebyl narozdíl od ostatních aplikací nalezen.

5.3.2 SPARQL konektory

V rámci *Power BI* žádný přímý SPARQL konektor s podporou RDF dat neexistuje *(k 8.12.2023)*. Power BI poskytuje možnost získání dat z webového zdroje pomocí GET

metody. SPARQL Endpoint, u kterého velikost dotazu nepřesahuje limit podporovaného prostoru v URL a který podporuje GET volání, lze dotazovat. V případě, že si uživatel nevystačí s GET dotazem, existuje možnost vytvoření POST požadavku v Power BI jazyku *Power Query M*, jehož konstrukce lze v rámci aplikace používat. Je i možnost použití skriptu jazyka Python, který je také použitelným zdrojem. HTTP a HTTPS požadavky vrací pouze čisté odpovědi ze SPARQL Endpointu, které je potřeba zpracovat do použitelné podoby, viz 5.3.1 RDF formáty.

V rámci *Looker Studio* existuje celá řada konektorů. Základní rozdělení činí Google konektory a partnerské konektory, kdy Google konektory jsou konektory přímo vyvíjené týmem Looker Studio. V rámci partnerských konektorů existuje konektor přímo pro SPARQL. Komunitní SPARQL konektor od uživatele očekává nastavení URL, SELECT dotaz, ale zároveň i namapování datových typů na jednotlivé proměnné z výsledku. Možnost nastavení autorizační metody a přihlašovacích údajů zde není k dispozici. Dotaz obsahující znak * není povolen. Kromě fakta, že si uživatel musí manuálně veškeré proměnné namapovat na datový typ, je konektor použitelný. Zajímavý příklad použití tohoto konektoru lze nalézt například v článku *Dashboards from SPARQL knowledge graphs using Looker Studio*[7].

V rámci *Tableau* pro SPAROL Endpoint ani RDF data žádný konektor nebo komunitní konektor neexistuje. Nebyla ani nalezena přímá podpora využití API volání, ke kterým je potřeba vytvořit webové konektory, jež musí splňovat pro využití v aplikaci specifické vlastnosti.

5.3.3 Využití middleware

Pokud chce uživatel*Power BI* nebo *Tableau* získat data ze SPARQL Endpointu, ale nehodlá se zabývat transformací, je cestou využití nějaké existující aplikace jakožto prostředníka. Pro daný problém lze využít OpenLink Virtuoso [15], které umožňuje pracovat se SPARQL dotazem jako s SQL výrazem nad databází, což je v obou aplikacích nativně podporováno. Další možností je využití existující webové služby, která nemusí být přímo SPARQL Endpoint, ale poskytuje rozhraní, ze kterého lze připravená data získat.

5.4 Porovnání transformací

5.4.1 Transformační obrazovka

V *Power BI* se po zvolení zdroje dat a úspěšném čtení těchto dat uživatel ocitne v části aplikace zaměřené na transformace dat. Před uživatelem se nachází tabulka

[7] Dashboards from SPARQL knowledge graphs using Looker Studio - https://blog.sparna.fr/2022/10/18/dashboards-from-sparql-knowledge-graphs-using-looker-studio-google-data-studio/

obsahující základní přehled získaných dat (dle použitých transformací).

V *Looker Studio* se po zvolení a úspěšném načtení zdroje dat uživatel ocitne v obrazovce, kde může vidět jednotlivé atributy, nikoliv však tabulku jejich hodnot, jako tomu bylo
v Power BI.

V *Tableau* je po získání dat zdroje uživatel v části aplikace zaměřené na transformace. Před uživatelem se nachází tabulka obsahující prvních 10 000 načtených hodnot. Je možné data řadit, přejmenovat sloupce, nebo je odstranit. Další transformace jsou možné až v rámci další obrazovky aplikace, pracovním sešitu (viz 5.5.1 Tableau), kde může transformace pokračovat.

Jednoznačně nejvhodnějším se jeví řešení Power BI, které rozděluje transformační a vizualizační část. Zároveň je ihned po získání dat možnost jejich prozkoumání v interaktivním tabulkovém přehledu.

5.4.2 Datové typy

Power BI se snaží samo rozpoznat datový typ sloupce na základě podmnožiny jeho hodnot (prvních 1000 záznamů). *Looker Studio* umožňuje pouze mauální nastavení datových typů sloupců. *Tableau* umí u některých zdrojů (např. SQL) automaticky přiřadit datový typ. U ostatních je nutné manuální nastavení tohoto datového typu.

Tabulka 5.1 porovnává datové typy, které tvoří souhrn přímo dostupných datových typů v porovnávaných aplikacích. Pokud mají datové typy v aplikacích různé označení, ale fungují stejně, je název pro porovnání v tabulce sjednocen. Datové typy v rámci analyzovaných aplikací se podobají.

Tabulka 5.1: Porovnání datových typů aplikací

Datový typ	Power BI	Looker Studio	Tableau
numerické			
- celé číslo	ano	ano	ano
- desetinné číslo	ano	ano	ano
- procento	ano	ano	ne
- doba trvání	ano	ano	ne
- měna	ne	ano	ne
datum a čas			
- datum a čas	ano	ano	ano
- datum	ano	ano	ano

(tabulka pokračuje na další stránce)

Tabulka 5.1 (*pokračování z předchozí stránky*)

Datový typ	Power BI	Looker Studio	Tableau
- čas	ano	ano	ne
- časové pásmo	ano	ano	ne
- rok	ano	ano	ano
- čtvrtletí	ano	ano	ne
text	ano	ano	ano
boolean	ano	ano	ano
adresa URL			
- adresa	ano	ano	ne
- obrázek	ano	ano	ano
geografické údaje			
- kontinent	ano	ano	ne
- země	ano	ano	ano
- dílčí územní celek země	ano	ano	ano
- město	ano	ano	ano
- PSČ	ano	ano	ano
- adresa	ano	ano	ano
- zeměpisná délka a šířka	ano	ano	ano

5.4.3 Filtrace hodnot

V *Power BI* lze nad jednotlivými sloupci provést filtrace dle konkrétních hodnot. Následně je možné použít i filtrace specifické pro odpovídající datový typ sloupce. Např. filtrace textu, čísel, datumu a času: *text* (je rovno, není rovno, začíná na, nezačíná na, končí na, nekončí na, obsahuje, neobsahuje), *čísla* (je rovno, není rovno, větší než, je větší než nebo rovno, menší než, je menší než nebo rovno, mezi), *datum a čas* (mezi, před, po).

Oproti Power BI v transformační obrazovce *Looker Studia* nebyla nalezena žádná možnost předpřipravit si hodnoty dat pomocí filtrací. Další filtrace dle datových typů jsou opět jako u Power BI.

Tableau umožňuje nastavení pravidel filtrů obdobných Power BI. Je zde rozlišena aplikace daných filtrů: na celý zdroj dat, po sloupci, po pravidlu, beze jména.

Filtrace dle datových typů se v rámci jednotlivých aplikací podobají. Filtrace dle hodnot a specifická filtrační pravidla dle datového typu jsou užitečné.

5.4.4 Základní statistiky sloupce

V *Power BI* a *Tableau* lze nad každým sloupcem zobrazit jeho základní statistiky a distribuci hodnot v podobě grafu. Nevýhodou je, že se vychází pouze z prvních 1000 načtených hodnot (10 000 u Tableau). V případě většího rozsahu dat tak nemusí mít informace velkou vypovídající hodnotu. Mohlo by být užitečné poskytnout volitelnou možnost získání statistik ze všech dostupných řádků.

V *Looker Studio* nebyly obdobné možnosti jako u ostatních aplikací pro statistiky sloupce nalezeny.

Zobrazení statistik sloupce v rámci transformační obrazovky se jeví jako velice užitečný nástroj. Rychlé souhrny odlehlých nebo převažujících hodnot dají lépe porozumět chování datového souboru. Nástroje Power BI a Looker Studio tento problém dobře řeší.

5.4.5 Operace

Power BI nabízí mnoho dalších transformačních operací. Zároveň umožňuje upravení získaných dat pomocí skriptu ve zmíněném jazyce M, Python nebo R. Základní transformační operace: *zachovat řádky* (dle filtračního kritéria), *odebrat řádky* (dle filtračního kritéria), *rozdělit sloupec* (oddělovačem, podle počtu znaků, podle pozic), *seskupit podle* (agregace: součet, průměr, medián, min, max, počet řádků, počet jedinečných řádků, všechny řádky), *transponovat* (řádky na sloupce, sloupce na řádky), *obrátit řádky* (změna pořadí), *odebrat duplicity*, *odebrat chyby*, *změnit datový typ*, *nahradit hodnoty*, *nahradit chyby*, *transformace hodnoty*.

Transformace *Looker Studio* poskytují možnost rychlého vytvoření polí metrik na základě hodnot sloupců. Kliknutím na datový sloupec z něj lze vytvořit dále použitelnou metriku (počet) nebo ho duplikovat. V rámci transformační obrazovky existuje možnost vytvoření výchozích agregací dle zvoleného datového typu atributu. Příklad agregací numerického datového typu: *žádná, celkem, průměr, počet, přesný počet, min, max, medián, směrodatná odchylka, odchylka*. Je dostupný výpočet pole pomocí vzorce, kdy si uživatel relativně intuitivně volí klikáním jednotlivá pole a vkládá mezi ně operátory. Možnosti transformačních operací a jejich množství jsou značně omezenější než v aplikaci Power BI.

V *Tableau* je s jednotlivými sloupci možné provádět operace, jejichž množství je bohatší než u aplikace Looker Studio: *vytvoření aliasů* (namapování hodnot sloupce na jiné), *vytvoření vypočteného pole* (dle vzorce), *seskupení hodnot*, *hierarchie dat*, *rozdělení sloupce, nastavené rozdělení, vytvoření datumu*, *konverze na míru* (následně může být použito jako řádky), *změna datového typu*.

Základními transformačními operacemi, které se vyskytují u všech aplikací jsou seskupování hodnot a jejich agregace. Dále jsou základem možnosti výpočtu pole pomocí vzorce a konverze hodnot nebo rozdělení sloupců. Nejbohatší množství transformací a jejich možností bylo u aplikace Power BI.

5.4.6 Uchování transformačních kroků

Protože v transformaci *Power BI* jde o práci s daty, u kterých se předpokládá, že se obsahově, nikoliv strukturálně, budou v čase měnit, je nutné uchovat provedené transformační kroky, které jsou aplikovány vždy při aktualizaci dat. V uživatelském rozhraní je zobrazen použitý postup, za kterým se pro jednotlivé kroky skrývá konkrétní volání transformační funkce. Jednotlivé kroky transformací lze libovolně odstraňovat, měnit jejich pořadí, nebo na ně přecházet. Aplikace sama zpracuje transformace tak, aby se data nacházela v odpovídajícím stavu.

Oproti Power BI v *Looker Studiu* a *Tableau* žádné zobrazení použitých transformací nebo jejich editace nebylo nalezeno.

Uchování, úprava a zobrazení transformačních kroků, které implementuje aplikace Power BI, se jeví jako velice užitečné pro uživatelův přehled, jak bylo se zdrojem dat pracováno, a možnou snadnou správu tohoto procesu.

5.4.7 Modelování dat

V *Power BI* lze vytvářet datové modely, které obsahují vztahy mezi tabulkami. To umožňuje propojení dat a usnadňuje analýzu v rámci různých tabulek, kdy aplikace ví, že se například jedná o stejný klíč. Vytvoření hierarchií je užitečné pro analýzu dat na různých úrovních. Lze vytvářet skupiny, aby mohla být data seskupována podle určitých kritérií. Příkladem je datum, kdy získaná hierarchie může rozdělit data na rok, měsíc, den apod.

V rámci *Tableau* existuje obdobná možnost vytvoření hierarchie sloupce jako u Power BI, kdy se může jednat o hierarchii ze záznamu JSON-LD souboru, kde se může rozlišovat pro jeden záznam ve sloupci type a value. Další operace modelování vztahů mezi daty je propojení zdrojů dat podle klíče. Není tady dostupné pokročilé modelování relací mezi více tabulkami, jako v Power BI.

Oproti Power BI a Tableau v *Looker Studiu* pokročilé možnosti modelování dat nebyly nalezeny.

Základní možností modelování dat se jeví možnost spojování zdrojů dat. Hierarchie dat je dalším prvkem, který může intuitivně řešit výskyt složitějších dat v podobě objektů nebo polí.

5.5 Porovnání dashboardu

5.5.1 Rozložení

Pokud má uživatel *Power BI* nebo *Looker Studia* připravena data, přechází do obrazovky dashboardů, kde se již vytvářejí vizualizace. Vizuály jsou seskupené do jednotlivých stránek a nabízí se vytvoření rozložení dashboardů pro PC a pro mobilní zařízení, kdy jsou tyto dvě možnosti odděleny. Každá stránka představuje jeden dashboard. Samotné pozice vizuálů a jejich velikostí se určují myší roztažením objektu vizuálu. Jsou automaticky zobrazovány zarovnávací osy pro přichycení k určitému bodu.

Jeden soubor v *Tableau* je jednou pracovní knihou (workbook). Tento workbook pak obsahuje sadu pracovních sešitů (worksheetů). Worksheet představuje jeden vizuál, s nastavením jeho polí do řádků, sloupců. Výběr vizuálu, který představuje worksheet, je možný dle nastaveného mapování sloupců a řádků, kdy jsou zobrazeny jen vizuály, které s daným počtem měr a dimenzí lze použít. Dashboard je složen z několika worsheetů, které jsou doplněny o nastavení rozložení. User story je prezentací dashboardů, která má vyjádřit informaci.

Stránkové rozložení dashboardů, s množstvím libovolně rozmístitelných vizuálů, které používají různé datové zdroje z dashboardu, a jsou použity u Power BI a Looker Studia, se jeví jako velice uživatelsky přívětivé a efektivní.

5.5.2 Vizuály

V *Power BI* a *Looker Studio* jsou k dispozici všechny zmíněné vizuály z části 4.3.2, a navíc je možnost importování vizuálů, které jsou vytvářeny komunitou. Možnosti jsou tedy téměř neomezené.

Ve worksheetu *Tableau* lze použít veškeré základní vizuály zmíněné v části 4.3.2.

Množství vizuálů k použití se jeví rozsáhlejší pro aplikace Power BI a Looker Studio.

5.5.3 Vlastní vizuály

Pokud není požadovaný vizuál nalezen, existuje možnost vytvoření vlastního vizuálu pro Power BI pomocí jazyka DAX, pro Looker Studio pomocí jazyka Javascript s použitím vývojářských knihoven a pro Tableau s využitím Tableau Extensions API.

5.5.4 Ovládací prvky

*Power BI*umožňuje využití ovládacích prvků. Například jsou k dispozici tlačítka, která mají přiřazenou nastavitelnou funkcionalitu, jako je přechod mezi stránkami, navigace na záložku, přepnutí filtrů apod.

Podobně jako v Power BI, *Looker Studio* umožňuje využití ovládacích prvků, jako jsou tlačítka, která mohou být přiřazena různým funkcionalitám, například přechodu mezi stránkami, navigaci na záložku, přepnutí filtrů atd. Ovládací prvky v Looker Studio se ale hned po vytvoření nemusí obtížně nastavovat, automaticky jsou předem na očekávanou funkcionalitu nastaveny (např. filtrace s ovládacím prvkem období). Příklad ovládacích prvků Looker Studia: *rozbalovací seznam, seznam s pevnou velikostí, zadávací pole, rozšířený filtr, posuvník, zaškrtávací políčko, předvolený filtr, ovládací prvek období, kontrola nad daty, ovládací prvek dimenzí, tlačítko*.

Ovládací prvky se v *Tableau* nenacházejí kvůli povaze daných pracovních sešitů. Ty dokonce obsahují průvodce nastavení, který dle aktuálního nastavení zobrazuje odlehlé hodnoty, povahu dat a navrhuje jaké vizualizace použít.

Uživatelsky přívětivé ovládací prvky se jevily být k dispozici v aplikaci Looker Studio, kde jejich nastavení bylo rovnou přednastavené, nebo velice intuitivně upravitelné a dobře aplikovatelné na zvolenou stránku dashboardu.

5.5.5 Grafické prvky

V *Power BI* jsou mimo funkční ovládací prvky a vizuály k dispozici i grafické prvky, jako textové pole, tvary nebo obrázky. To umožňuje dashboard graficky přizpůsobit dle potřeb.

Kromě funkcionálních ovládacích prvků a vizuálů nabízí *Looker Studio* grafické prvky jako textová pole, tvary nebo obrázky. Každý vizuál nebo ovládací prvek, kromě samotného nastavení, přináší i záložku styl, kde lze detailně upravit jeho vzhled. To umožňuje uživatelům graficky přizpůsobit dashboard podle svých potřeb a preferencí, což přispívá k možnosti esteticky přitažlivého a přehledného zobrazení dat. Oproti Power BI je zde možnost přizpůsobení barevného motivu celého dashboardu, který je pak použit na všech vizuálech, což je snadnější, ale méně detailní, možnost úpravy vzhledu dashboardu.

V *Tableau* jde použité komponenty rozsáhle vzhledově přizpůsobovat. Samostatné grafické prvky se v aplikaci pro použití nenachází. Vše vykreslené v dashboardech je zkrátka worksheet.

Grafické přizpůsobení a možnosti grafických prvků se jeví jako nejlepší v aplikaci Looker Studio. Kombinace přizpůsobení barevného motivu celého dashboardu a i přizpůsobení vizuálů jsou rozsáhlé.

5.6 Porovnání interakce s vizuálem

5.6.1 Přiřazení dat vizuálům

V *Power BI* a *Looker Studiu* mají jednotlivé vizuály své atributy, kterým se přiřadí konkrétní sloupce ze zdroje dat. Sloupec z panelu zdrojů dat se přetáhne myší na příslušné pole atributu ve vizuálu. Vizuál tabulka například obsahuje atribut sloupce, graf atribut Osa X, Osa Y a popisky dat. Jednotlivá pole u vizuálu nabízejí možnosti agregací, kdy agregace nastává až přímo ve vizuálu.

Interakce v *Tableau* funguje přetažením sloupce do polí dimenzí a měr. Pro každý vizuál funguje přiřazení stejně, vychází ze stejného nastavení, jen se mění vykreslený vizuál.

Pro přiřazení dat vizuálům používají všechny aplikace tažení myši na příslušné pole nastavení. Aplikace Tableau ukazuje, že není potřeba pro rozdílné vizuály měnit nastavovaná pole, která jsou zde pouze rozlišována na dvě osy (míry a dimenze), což se jeví jako použitelné zjednodušení.

5.6.2 Filtrace

Vizuály v *Power BI* zpravidla umožňují nastavení vlastních filtrů pro jednotlivé sloupce, které ovlivní jen daný vizuál. Je možnost filtrace dat pro celou stránku, kdy jsou ovlivněny všechny vizuály na dané stránce.

Filtrace v *Looker Studio* může být aplikována nad celou sestavou nebo nad jednotlivými vizuály. Uživatelé mohou používat pojmenovaná filtrační pravidla, která mohou být zapnuta nebo vypnuta pro jednotlivé vizuály. Tato filtrační pravidla umožňují uživatelům dynamicky ovlivňovat data zobrazená ve vizuálech a provádět interaktivní analýzy.

Jak bylo zmíněno, filtrace v *Tableau* probíhá na úrovni jednoho worksheetu (vizuálu).

Možnosti nastavení filtrace zvlášť na vizuál a celý dashboard se pro aplikace opakují a jeví se efektivní a uživatelsky přívětivé. Možnosti pojmenování filtrovacích pravidel a jejich uložení v aplikaci Looker Studio jsou zajímavé.

5.6.3 Zoom and filter

Kliknutí na konkrétní záznam v tabulce, bod v grafu nebo například kategorii ve výsečovém grafu, ovlivní všechny vizuály na stránce a tento filtr je na všechny použit, což umožňuje intuitivní průzkum *zoom and filter*. Křížové filtrování lze na vizuálu aktivovat či deaktivovat.

5.6.4 Details on demand

Power BI umožňuje i intuitivně vytvořit dashboard pro detailní pohled na určitá data. Po vytvoření této stránky, která odpovídá detailu nějaké třídy nebo hodnoty, lze tuto stránku přiřadit k určitému datovému poli jako podrobnou analýzu. Po přechodu k podrobné analýze jsou zachovány filtry z předchozího výběru, zejména pak vybrání této položky, na základě čehož by měl být zobrazen vizuál, kde lze zjistit bližší informace právě o této jedné entitě.

V *Looker Studiu* a *Tableau* se detailnější informace obvykle zobrazují při najetí myší na určitou část vizuálu.

Možnosti nastavení detailního pohledu jako stránky dashboardu u aplikace Power BI jasně převyšují možnosti ostatních aplikací.

5.7 Porovnání sdílení

V *Power BI* je sdílení reportů dosažitelné několika kliknutími přímo z aplikace Power BI Desktop. Následně je uživatel přesměrován na web Power BI, kde se již nachází publikovaný dashboard, z nějž lze snadno získat HTML iframe, který lze sdílet dle potřeb.

Uživatelé *Looker Studia* mohou publikovat své dashboardy a vizualizace pomocí funkce *publikování*. Tato funkce umožňuje uživatelům sdílet své dashboardy s ostatními uživateli nebo s externími osobami. Při publikování je možné nastavit různá oprávnění a omezení, jako jsou například přístupová práva, možnosti editace a zobrazení. Uživatelé mohou generovat odkazy a vkládat dashboardy do webových stránek nebo aplikací.

Vytvořené dashboardy a user stories v *Tableau* lze sdílet pomocí Tableau Server nebo Tableau Cloud.

Každá z aplikací řeší sdílení dashboardu jeho publikováním do webové služby. U aplikace Looker Studio je výhodou, že uživatel nemusí přecházet na jinou platformu, než kde upravoval dashboardy a zdroje dat.

5.8 Knihovny pro webové vizualizace dat

Následně budou porovnány některé z hlavních knihoven pro vizualizaci dat ve webových aplikacích. Knihovny byly vybrány na základě vlastního průzkumu oblíbených nástrojů a inspirace z bakalářské práce *Jakuba Hrušovského* [29].

5.8.1 Popis knihoven

Google Charts

Google Charts je JavaScriptová knihovna pro vizualizaci dat, která vizualizace vytváří pomocí SVG. Navíc knihovna poskytuje i možnost využití jejího API, což umožňuje používání této knihovny při komplexnějších operacích. Za pomoci knihovny je možné vytvářet grafy, včetně sloupcových, liniových, histogramů, scatter plotů a dalších. Jedná se o knihovnu společnosti Google, která měla své počátky v roce 2007 a její poslední aktualizace *verze 52* proběhla v dubnu 2024 (k 9. 5. 2024). To přináší především jistotu, spolehlivost a dobrou dokumentaci. Jednou z aplikací, která knihovnu používá, je například známý SPARQL klient *YasGUI*. [30]

D3.js

D3 (*Data-Driven Documents*) je JavaScriptová knihovna určená pro manipulaci s dokumenty na základě dat, která využívá HTML, CSS a SVG. Knihovna poskytuje nástroje pro selekci, manipulaci a transformaci DOM (*document object model*). Klade důraz na deklarativní přístup k tvorbě vizualizací. Uživatelé mohou popsat vzhled a chování vizualizace pomocí datově řízených transformací, které jsou aplikovány na objekty. Tímto způsobem lze propojit data s vizuálními prvky a vytvářet reaktivní vizualizace, které automaticky reagují na změny dat. D3 umožňuje vytvářet vlastní vizuální prvky a interakce, je poskytováno mnoho nástrojů a metod pro práci s grafikou, animacemi, změnou atributů a řízením událostí. Vizuály lze podrobně přizpůsobit dle specifických požadavků. Knihovna je vyvíjena open-source a její vývoj začal již v roce 2011. Na svém githubu[8] má knihovna (k 9.5. 2024) 134 tisíc přispěvatelů a 108 tisíc hvězd, což z ní dělá vizualizační knihovnu jazyka JavaScript s nejvíce hvězdami. Knihovna je využívána 355 tisíci repozitáři a poslední zveřejněná *verze 7.9.0* byla vydána v březnu 2024 (k. 9. 5. 2024). [31]

Chart.js

Chart.js je JavaScriptová knihovna pro tvorbu interaktivních a esteticky příjemných vizualizací. Podporuje různé typy grafů, včetně sloupcových, spojnicových, koláčových, paprskových a mnoha dalších komunitně vytvářených grafů. Uživatel může definovat data, popsat styly, nastavit osy a přidávat popisky. Pokud uživatel neupraví data předaná do vizualizace, knihovna využije svého předem nastaveného vzhledu pro danou vizualizaci, která zahrnuje i výchozí animace. Responzivní design umožňuje, aby se vizualizace automaticky přizpůsobovaly velikostím a rozlišením obrazovky. Knihovna zahrnuje možnosti interakce s jednotlivými body vizualizace, což umožňuje získání podrobností, nebo přepínání mezi sadami dat. Chart.js vykresluje vizualizace pomocí *Canvas*, což znemožňuje využití CSS stylů, ale pomáhá

[8] D3.js github - https://github.com/d3/d3

optimalizovat vizualizaci velkých datových sad a komplexních vizualizací. Knihovna nabízí mnoho pluginů a rozšíření, která umožňují přidávat další funkcionality a vlastnosti do grafů. Na svém githubu[9] má knihovna (k 9.5. 2024) 484 tisíc přispěvatelů a 1.9 tisíc hvězd. První verze knihovny vyšla v roce 2015, nyní je využívána 933 tisíci repozitáři a poslední zveřejněná *verze 4.4.2* byla vydána v únoru 2024 (k. 9. 5. 2024).

5.8.2 Porovnání knihoven

Google Charts

Dle [29] umožňuje knihovna jednoduché a velmi přehledné vizuální zpracování pro všechny základní typy podporovaných typů grafů. Konfigurace těchto grafů je oddělena do samostatného objektu, což je dobrý způsob využití přizpůsobení v odlišných cyklech života grafu. Jedná se o knihovnu, která nabízí dobrý poměr přizpůsobení ku složitosti pro jednoduché i částečně pokročilé grafy.

D3.js

Dle [29] má knihovna větší komplexnost a její konfigurace je taktéž nejkomplikovanější. Podporuje vytváření jednoduchých i komplexních grafů. Umožňuje sice přizpůsobení téměř libovolným způsobem, ale zakomponování tohoto přizpůsobení v rámci komplexnějšího typu grafu je velmi složité.

Chart.js

Obtížnost konfigurace vizuálů je dle [29] podobná knihovně Google Charts. ChartJS oproti této knihovně poskytuje možnost vizualizace i několika pokročilých grafů. Konfiguraci grafů je obdobně možné oddělit od samotného grafu, ale knihovna v základu obsahuje méně možností přizpůsobení než Google Charts. Po přidání rozšíření je úroveň přizpůsobení obdobná. Práce s knihovnou byla označena jako jedna z méně složitých ale oblíbených, čemuž odpovídá i rozsáhlý počet repozitářů na Githubu, které knihovnu využívají.

[9] Chart.js github - https://github.com/chartjs/Chart.js

Návrh metody zpracování a vizualizace dat

6

Pro jednotlivé procesy zpracování dat dle členění v části 4 budou uvedeny možnosti pro data ze SPARQL Endpointu a bude ověřena jejich reálná použitelnost. Fáze transformace bude rozdělena na automatickou transformaci (po získání dat) a manuální transformaci dat (transformace dat uživatelem).

6.1 Získání dat

Základním předpokládaným zdrojem dat je odpověď na SPARQL dotaz nebo jinak získaná data ve formátu RDF. Pro získání odpovědi na dotaz je potřeba znát adresu endpointu, na který se má dotaz odeslat, řetězec samotného dotazu a případně i autorizační údaje v případě, že se jedná o soukromý endpoint. Druhou možností je zdroj dat, který již představuje odpověď z nějakého SPARQL dotazu nebo jinak získaná data v jednom z RDF formátů.

Zdroj dat: SPARQL dotaz

Metoda by před samotným provedením dotazu měla ověřit, zda se vůbec jedná o SPARQL dotaz se správnou syntaxí. SPARQL Endpointů napříč webem existuje již mnoho. Provoz SPARQL Endpointu je ale nákladný a v případě práce s ním je nutné nejprve ověřit, zda je vůbec v provozu a reaguje na SPARQL dotazy v očekávaném formátu. Pro ověření funkcionality endpointu postačí využití jednoduchého SPARQL dotazu:

```
1 SELECT * WHERE { ?s ?p ?o } LIMIT 1
```

Zdrojový kód 6.1: SPARQL dotaz: ověření funkcionality endpointu

V případě, že je navrácen odpovídající výsledek, SPARQL Endpoint funguje. Dalším krokem je zjištění verze daného SPARQL Endpointu. Stejně jako v minulé části postačí jednoduchý dotaz, který obsahuje konstrukce z jazyka SPARQL 1.1.

```
SELECT (COUNT(*) as ?c) WHERE {
    SELECT * WHERE { ?s ?p ?o } LIMIT 1
}
```

Zdrojový kód 6.2: SPARQL dotaz: ověření verze endpointu

V případě, že je vrácena řádná odpověď i na komplikovanější konstrukci obsahující zanořený dotaz a agregační funkci, předpokládá se, že endpoint podporuje SPARQL 1.1.

Zdroj dat: RDF soubor

Je-li zdrojem přímo soubor v RDF formátu, je potřeba, aby metoda uměla pracovat stejným způsobem s formáty z části 2.3. Dalším faktorem v případě použití vstupního souboru může být neznalost dotazu, adresy a údajů endpointu, ze kterého byla data získána. To může zkomplikovat získávání metadat k těmto datům a nebude možné plně využít potenciál navrhované aplikace.

Získání metadat pro hodnoty v datech

Pro získání metadat je nutná znalost adresy endpointu nebo je nutné mít potřebná data ve vlastním RDF úložišti. Data ontologií a slovníků nemusí být dostupná přímo na endpointu, ze kterého se získávají data. Očekává se, že vlastní úložiště, kam budou požadované ontologie nahrány, bude pro získání metadat vhodné. Pro dotazy na metadata byl vytvořen univerzální dotaz, do kterého je vnořeno pole subjektů (IRI), jejichž metadata jsou získávána. Dále je možnost zadání konkrétních predikátů. Pokud zůstanou predikáty nevyplněny, jsou navráceny všechny atributy těchto subjektů. Optional blok s jazykovými filtry je použit pouze tehdy, pokud je jazyk přítomen. V tom případě je zvoleno z množiny předvolených jazyků (cs, en). Formát SPARQL dotazu na získání metadat:

```
SELECT DISTINCT ?subject ?predicate
                (SAMPLE(?value) AS ?firstValue)
WHERE {
    VALUES ?subject {
      <http://www.wikidata.org/entity/Q204933>
      <http://www.wikidata.org/entity/Q210392>
      }
      VALUES ?predicate {rdf:type rdfs:range rdfs:domain}
      ?subject ?predicate ?value
      OPTIONAL {
          FILTER (LANG(?value) = "en" || LANG(?value) = "cs")
      }
}
GROUP BY ?subject ?predicate
```

Zdrojový kód 6.3: SPARQL dotaz: získání metadat subjektu/subjektů

V případě potřeby většího seznamu subjektů a predikátů nebo složitějšího dotazu, je vhodné dotazy odesílat dávkově po menším množství (např. po 1000 subjektech). U složitějších dotazů některé endpointy odpověď nemusí vrátit a může nastat chyba vypršení časového limitu.

Periodické obnovování dat a metadat

Analyzované aplikace běžně provádí periodickou aktualizaci zdrojů dat. U zdrojů s vyplněným SPARQL dotazem přichází v úvahu nastavení periodického aktualizování dat a případně i jejich metadat. Vhodná perioda aktualizace se může lišit dle povahy dotazu a odpovídajících dat, dále může být brána v potaz i doba nižšího vytížení dotazovaných endpointů, případně i vytížení aplikace, která by aktualizovaná data následně musela zpracovávat. Pokud je zdrojem dat vstupní soubor, je jen na uživateli tento soubor aktualizovat. Předzpracování pak bude prováděno vždy po poskytnutí nového souboru.

6.2 Automatická transformace dat

Po úspěšném získání dat následuje část automatické transformace, která se pokouší získaná data předzpracovat a získat co největší množství relevantních informací pro další práci s těmito daty. Práce bude uvažovat každou proměnnou z dotazu (*head, vars viz E*) jako sloupec, kdy přezpracování povahy dat bude probíhat právě na úrovni jednotlivých sloupců.

6.2.1 Analýza pozice proměnných z dotazu

V případě, že je k dispozici vstupní dotaz, dají se potenciálně relevantní informace o sloupcích automaticky zpracovat přímo z textu dotazu. Příklad bude uveden na dotazu:

```
SELECT
  ?age ?illness (COUNT(?person) AS ?count)
WHERE
  {
    {
    SELECT
      ?person (SAMPLE(?age) AS ?age) ?illness
    WHERE
      {
        ?person   wdt:P509 ?illness     ;
                  wdt:P31 wd:Q5         .

        OPTIONAL { ?person wdt:P570 ?d }

```

```
        ?person    wdt:P569 ?dob         ;
                   wdt:P570 ?dod            .

        BIND(YEAR(?dod) - YEAR(?dob) AS ?age)
      }
    GROUP BY
      ?person ?illness
    }
  }
GROUP BY ?age ?illness
```

Zdrojový kód 6.4: SPARQL dotaz: počet výskytu nemocí dle věku (wikidata.org)

Proměnná sloupce se může vyskytovat na pozici *subjektu*, *predikátu* i *objektu*. V každé pozici jsou zároveň známé hodnoty dvou ostatních pozic. Jednotlivé scénáře budou následně probrány a bude prozkoumáno, co se dá z daného scénáře vyvodit. Pozice proměnné *?illness* z dotazu 6.4:

```
nalezeno na pozici subjekt: NE

nalezeno na pozici predikátu: NE

nalezeno na pozici objektu: ANO
    subjekt: ?person
    predikát: wdt:P509
```

Zdrojový kód 6.5: Zpracované pozice proměnné ?illness

V případě, že je sloupec rozpoznán na pozici objektu, má relevanci informace o jeho predikátu. Pokud je nalezen jednoznačný statický predikát (nikoliv proměnná), lze se pomocí upraveného dotazu 6.4 pokusit získat jeho metadata. Metadata přímo pro *wdt:P509* nebyla nalezena. Odpovídající property je na wikidata totiž pod zdrojem *wd:P509*:

```
rdf:type  wikibase:Property
rdfs:label  "příčina úmrtí"@cs
schema:description    "underlying or immediate cause of
death. Underlying cause (e.g. car accident, stomach cancer)
preferred. Use 'manner of death' (P1196) for broadest
category, e.g. natural causes, accident, homicide,
suicide"@en
```

Zdrojový kód 6.6: Ukázka získaných metadat pro wd:P509

Pro tento zdroj byl nalezen typ (*rdf:type*), popisek (*schema:description*) i označení (*rdfs:label*). Pokus o získání metadat jiného možného predikátu *dbo:birthPlace* endpointu *dbpedia.org*:

```
rdf:type  rdf:Property, owl:ObjectProperty
rdfs:label  "birth place"@en
```

```
rdfs:range   dbo:Place
rdfs:domain    dbo:Animal
```

Zdrojový kód 6.7: Ukázka získaných metadat pro *dbo:birthPlace*

V tomto případě již nastala vhodnější situace. Mimo label je zde i vlastnost *rdfs:range*, která určuje typ objektu za tímto predikátem (zkoumaného sloupce). Vlastnost *rdfs:domain* říká, jaké třídě vlastnost náleží. To se může hodit v případě, že je zpracovávaný sloupec na pozici subjektu a je nalezen statický predikát. Proběhlo otestování i zpracování sloupce na pozici predikát, které žádné jednoznačné informace nepřineslo. Jediné, co z toho vyplývá, je, že se jedná o IRI hodnotu. Dále byly vyzkoušeny i dotazy na nestatické predikáty (nalezený predikát byl hodnota jiného sloupce), jejichž vyhodnocování bylo příliš časově náročné a nebyly dále uvažovány.

Obecně lze říci, že získání metadat z pozice proměnné v dotazu nepřináši vždy relevantní informace. Pokud nastane případ, kdy se podaří získat popisek nebo *rdfs:range* a *rdfs:domain* predikátu, může být výsledkem automaticky získaný název a popis sloupce nebo informace o předpokládaném datovém typu hodnot ve sloupci. Třídy *rdfs:range* a *rdfs:domain* property však o konkrétních hodnotách, které se nachází ve sloupci mnoho nemusí prozradit, protože se může jednat o třídy příliš hierarchicky vzdálené instanci třídy v daném sloupci (u IRI tříd). Pro upřesnění povahy sloupce je proto vhodné využít i další způsoby.

6.2.2 Získání statistik sloupce

Jak bylo zmíněno v kapitole E, data ze SPARQL Endpointu obsahují následující pole: *pokaždé*: type, value; *volitelné*: datatype, xml:lang. S touto znalostí přichází možnost spočítání statistik jednotlivých typu polí pro každý sloupec. Musí být brána v potaz i varianta, kdy daný sloupec v řádku neexistuje. V takovém případě bude uvažováno, že se jedná o záznam, který má hodnotu type = null (v úvahu by připadal i bnode, ale ten by byl v datech přímo uveden a z hlediska principů sémantického webu, by se jednalo o chybu). Příklad statistik sloupce *?illness*, který obsahuje pouze IRI hodnoty, z dotazu 6.4:

```
spočteno řádků: 17 164
prázdné řádky: 0
počet IRI: 17 164
počet literálů: 0
počet prázdných uzlů: 0
počet dle hodnoty (value): NE
počet dle datového typu (datatype): NE
počet dle třídy (class):
    počet http://www.wikidata.org/entity/Q204933: 163
    počet http://www.wikidata.org/entity/Q210392: 15
    apod.
```

```
počet dle jazyka: NE
```

Zdrojový kód 6.8: Zpracované statistiky proměnné ?illness

Příklad statistik sloupce *?age*, který obsahuje pouze hodnoty literálů, pro dotaz 6.4:

```
spočteno řádků: 17 164
prázdné řádky: 0
počet IRI: 0
počet literálů: 17 164
počet prázdných uzlů: 0
počet dle hodnoty (value):
    počet 62: 271
    počet 59: 266
    apod.
počet dle datového typu (datatype):
    počet http://www.w3.org/2001/XMLSchema#integer: 17 164
počet dle třídy (class): NE
počet dle jazyka: NE
```

Zdrojový kód 6.9: Zpracované statistiky proměnné ?age

Stav, kde se jedná o sloupec pouze s hodnotami literálů nebo hodnotami IRI, nemusí nastat vždy. Spočítané statistiky umožňují rozhodnout dle četnosti o převažujícím typu sloupce.

6.2.3 Předzpracování - LITERÁL

Ve statistikách převažují hodnoty literálu s určitým datovým typem. Zjištění základního datového typu tohoto sloupce se může jevit v některých případech velice intuitivní, protože je obsaženo přímo v odpovědi. To ale platí pouze v případě, že je uveden známý datový typ, např. ze základního slovníku datových typů XMLSchema uvedeného v 2.4.1. Přehled nejběžnějších XSD datových typů je uveden v tabulce 6.2.

Tabulka 6.1: Ukázka XSD datových typů

Typ	IRI	Rozsah hodnot
logické	xsd:boolean	true, false
číselné	xsd:byte	8 bitový integer
číselné	xsd:int	32 bitový integer
číselné	xsd:long	64 bitový integer
číselné	xsd:float	32 bitový float
číselné	xsd:double	64 bitový float

(tabulka pokračuje na další stránce)

Tabulka 6.1 (*pokračování z předchozí stránky*)

Typ	IRI	Rozsah hodnot
textové	xsd:string	řetězec
časové	xsd:date	YYYY-MM-DD
časové	xsd:dateTime	YYYY-MM-DDTHH:MM:SS

Tyto třídy jsou pro použití na obecných dotazech moc detailní. Práce proto zavede datové typy, do kterých budou užší třídy datových typů následně přiřazeny:

Tabulka 6.2: Členění datových typů literálů dle práce

Typ	Označení
řetězec	*String*
číslo	*Number*
datum a čas	*DateTime*
booleaovská hodnota	*Boolean*

V případě, že aplikace datový typ nerozpozná, je nutná existence možnosti přiřazení známého datového typu. Pro nerozpoznaný datový typ bude přiřazen výchozí datový typ (např. řetězec) a uživateli bude oznámeno, že datový typ nebyl rozpoznán. Metoda zpracování by měla podporovat možnost mapování tříd datových typů na obecný datový typ, k čemuž může být hned přidáno nastavení vykreslení daného sloupce vytvořeného z třídy datového typu (viz 6.3).

6.2.4 Předzpracování - IRI

Na každý záznam bude pohlíženo jako na zdroj (*IRI = Internationalized Resource Identifier*). Ten může být číselníkem, kategorií nebo jiným libovolným typem zdroje, to však odhalí až samotná povaha IRI hodnot daného sloupce.

Vzorek vlastností

Prvním logickým krokem je získání typických vlastností zdrojů ve sloupci pomocí upraveného dotazu 6.3. Získání ukázkových vlastností zdrojů bude vykonáno nad vzorkem dat (například deset nejčetnějších IRI hodnot) ze sloupce *?illness* z dotazu 6.4:

```
rdfs:label: "Poprava ukamenováním"@cs
wikibase:statements: "90"
wikibase:identifiers: "49"
wdt:P1036: "616"
apod.
```

Zdrojový kód 6.10: Vzorek vlastností IRI sloupce ?illness

Získání vzorku dat jasně ukáže, jaké vlastnosti je možné z různých zdrojů ve sloupci získat. Zároveň bude jasné, zda vůbec požadované vlastnosti jsou na endpointu a RDF úložišti dostupné (v případě prázdné odpovědi). U některých vlastností je z hodnoty predikátu význam zřetelný, ale u vlastnosti jako *wdt:P1036* by bylo pro zobrazení vhodné použít v případě dostupnosti jejich slovní reprezentací (např. *rdfs:label*).

Načtení běžných vlastností

Očekává se, že zobrazení IRI sloupce by bylo uživatelsky nepřívětivé. Získání *rdfs:label* pro každou IRI hodnotu se považuje za samozřejmý předpoklad. Dalším předpokladem je možnost definování seznamu vlastností (predikátů), které se mají pro každou IRI hodnotu získat. Složitost dotazů je závislá na počtu rozdílných IRI hodnot ve sloupci. Metoda uvažuje předpoklad dotazu 6.4 a získání metadat bude probíhat dávkově.

6.2.5 Diagram automatické transformace

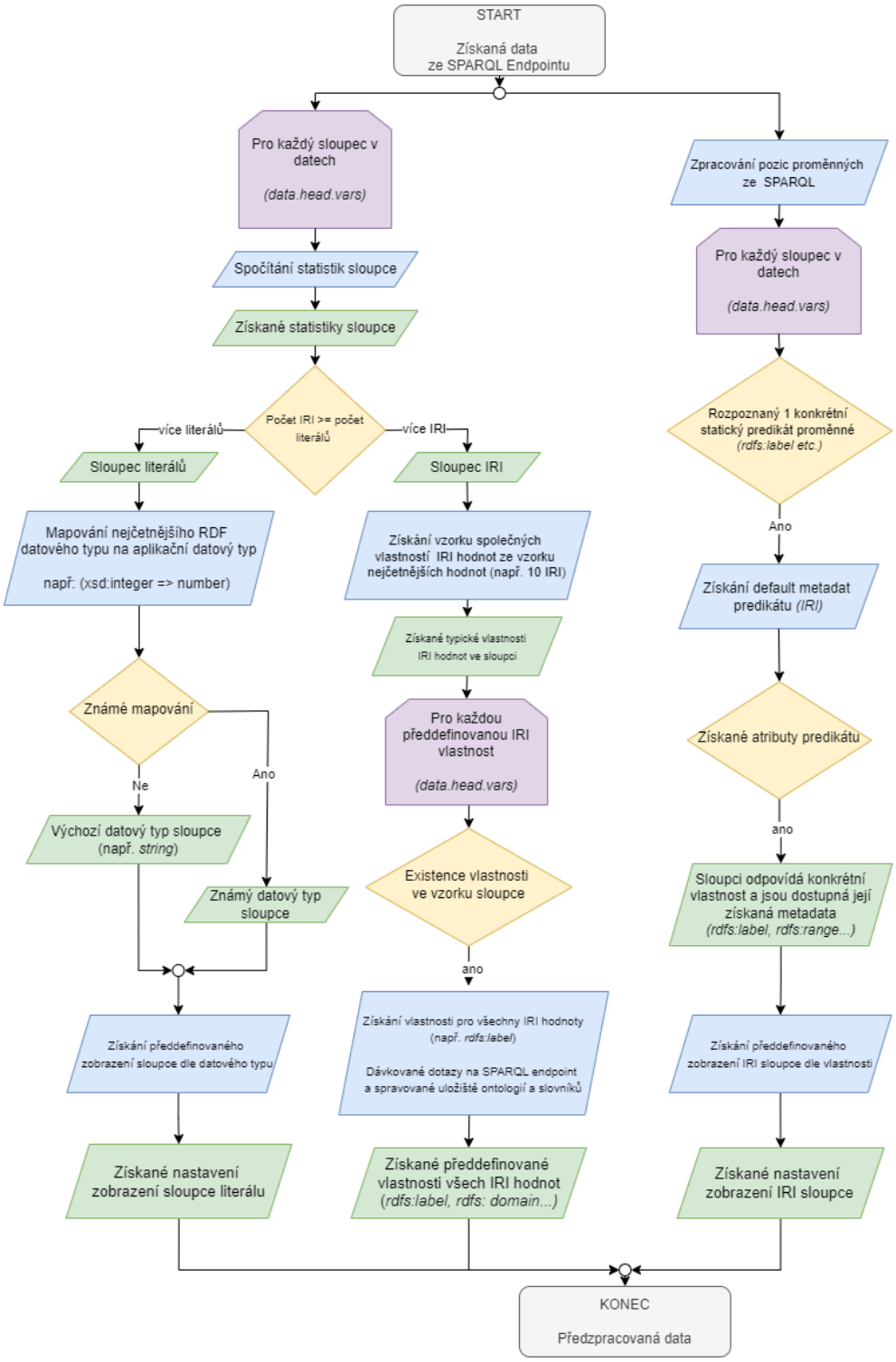
Využitím popsaných metod automatické transformace vzniká proces automatického zpracování sloupců, který je znázorněn na obrázku 6.1.

6.3 Manuální transformace dat

Za manuální transformaci dat je považováno vše, co musí uživatel učinit manuálně nad daty, která jsou již získaná a automaticky předzpracovaná (viz 6.2). Takové transformační kroky pak mohou probíhat automaticky při novém načtení těchto dat.

Zobrazení hodnot

Obecné: Možnou transformací sloupce je úprava názvu sloupce, kdy bude nejprve použit název z dotazu nebo ze získaných vlastností z pozic proměnné. Jelikož jsou vstupní data uvažována nejprve jako tabulka, je možné přizpůsobit i její zobrazení. Sloupec různých dat může vyžadovat rozdílnou šířku a zarovnání svého obsahu (střed, doprava, doleva). Ať se bude jednat o jakýkoliv datový typ,



Obrázek 6.1: Diagram automatické transformace RDF dat

vždy budou hodnoty zobrazeny v textové podobě (mimo například boolean, obrázek nebo geografické údaje, kde lze uvažovat i jiné zobrazení). Textu lze nastavit font, barvu, pozadí nebo velikost. Dále přichází v úvahu možnost nastavení pravidel zvýraznění určitých hodnot (např. zelené pozadí pro hodnoty > 5).

IRI: U IRI sloupce je možnost zobrazení zvolené vlastnosti tohoto zdroje místo původní podoby. Pokud si uživatel přeje zobrazit původní formu IRI, nebo nemá k dispozici další vlastnosti zdroje, může zvolit mezi zkrácenou podobou (prefix) nebo zobrazením celého IRI.

Číslo: Zobrazení číselného datového typu přináší možnost nastavení počtu číslic zobrazených za desetinnou čárkou. Dále volbu znaku pro oddělení celého čísla od jeho desetinné části. U větších celočíselných hodnot lze určit oddělení číslic dle počtu a oddělovací znak (např. seskupeno po 3 číslicích, odděleno " ": 10 000 000)

Text: U textu je v případě delšího řetězce vhodná možnost nastavení maximálního počtu zobrazených znaků, případně určení podřetězce, který se má zobrazit (regex, číselné indexy znaků).

Datum a čas: Hodnoty datum a čas v RDF budou standardně nabývat rozsahu hodnot *xsd:date* a *xsd:dateTime*. Může ale nastat i nestandardní scénář, kdy může libovolná třída představovat hodnotu datum a čas. Pro obecné možnosti práce s tímto datovým typem bylo rozhodnuto uživatelské zadání vzoru vstupního a výstupního řetězce, které bude pro XSD datové typy implicitně známé. Vzor ale umožní přívětivé přizpůsobení zobrazovaných hodnot. Příklad použití:

```
Vstup: 2017-05-23T13:00:00+0200
- vzor zpracování datumu: %Y-%M-%dT%h:%m:%s:%u+%t
- vzor zobrazení datumu: %d.%M. %Y %hh
Výstup: 23.5. 2017 13h
```

Zdrojový kód 6.11: Ukázka použití vzoru pro zpracování hodnoty datumu

V případě potřeby řazení je nutné, aby aplikace dovedla na základě poskytnutého vzoru určit reálný datum a čas. Pro aplikaci vzorů je možné použít libovolné znaky, ale následující znaky jsou vyhrazeny pro účel zpracování reálné hodnoty datumu a času: *rok* (%Y), *měsíc* (%M), *den* (%d), *hodina* (%h), *minuta* (%m), *sekunda* (%s), *setiny* (%u), *časová zóna* (%Z).

Pro výstupní vzor přibývá i možnost ze zpracovaného datumu zobrazit čísla týdne v roce, zobrazení dne v týdnu nebo číslo kvartálu v roce: *týden v roce* (%W), *den v týdnu* (%w), *číslo kvartálu* (%Q).

Například hodnotu měsíce je pak možné zobrazit i textově. Zde bude možnost volby numerického, českého nebo anglického zobrazení hodnoty měsíce. Podobně může i číslo dne v týdnu být ve výstupním řetězci nastaveno numericky, s anglickým nebo českým názvem. Příklad výstupního vzoru a zobrazení hodnoty:

```
Vstup: 2017-05-23T13:00:00+0200
- vzor zobrazení datumu: %d. %M %Y %hh
- zobrazení měsíce: anglické
Výstup: 23. May 2017 13h
```

Zdrojový kód 6.12: Ukázka nastavení zobrazení hodnoty datumu

Pro řazení a seskupování bude použita zobrazená forma hodnoty datumu dle řetězce vzoru zobrazení.

Boolean: U booleovské hodnoty lze zvolit, jak se má zobrazit hodnota true a false. Například lze definovat řetězec pro hodnotu true a řetězec pro hodnotu false. Přichází v úvahu i možnost vykreslení pomocí grafického prvku zaškrtávacího pole.

Filtrace dat

Specifickou filtrací RDF dat je filtrace dle typu, datového typu, tříd a jazyka (viz 6.2.2). Dále už by se filtrace měly podobat běžným filtracím dle datového typu, jako například u aplikace Power BI viz 5.4.3. Filtrace IRI sloupce se může chovat jako filtrace řetězce s tím, že bude filtrována momentálně zobrazovaná hodnota (pokud je zobrazena určitá vlastnost IRI zdroje).

Transformace IRI sloupce

U IRI sloupce je možné zejména přidání určité vlastnosti do možností zobrazení daného sloupcem nebo rovnou přidání této vlastnosti jako nového sloupce.

Běžné transformace dle datového typu

Je možné použít další běžné transformace viz 5.4.5, s tím, že by se opět s IRI sloupcem zacházelo podobně jako s datovým typem řetězce.

6.4 Dashboard

Jedním z cílů práce bylo vytvoření nástroje, který poskytuje webovou platformu pro interaktivní vizualizaci dat ze SPARQL Endpointu. V návrhu dashboardu se lze nejlépe inspirovat u aplikace Looker Studio, která problematiku dashboardu a vizualizací řeší ve webové aplikaci velice intuitivně.

Rozložení

Pro prostor dashboardu bude obdobně jako u Looker Studia použitý vzor více stránek. Aplikace umožní i přetečení stránek, po kterém se objeví posuvník. Rozložení vizuálů a jejich velikosti se nechá na uživateli a bude mít pravomoc libovolně vizuály přesouvat po celé obrazovce a měnit jejich velikost dle potřeby. Byla by vhodná asistence zarovnání pomocí zobrazení zarovnávacích os, jako je tomu právě u Looker Studia, což uživatele pobídne k estetičtějšímu návrhu dashboardu.

Vizuály

Jelikož dashboardy již pracují s transformovanými daty, nijak se neliší ani povaha vizuálů, které jsou pro data vhodná. Budou použity běžné vizuály podobné ostatním aplikacím viz 4.5.

Nastavení vizuálu

IRI sloupce budou převážně představovat kategoriální proměnnou, v případě, že mají nastaveno zobrazení své číselné vlastnosti, můžou být i kvantitativní hodnotou. Vizuály budou mít zpravidla nastavení několika kategorických proměnných, dle kterých se budou kvantitativní hodnoty shlukovat. Kvantitativní hodnoty pak budou agregovány následujícími způsoby:

- *count:* Počet hodnot pro daný shluk.
- *average:* Průměrná hodnota pro daný shluk.
- *sum:* Suma hodnot pro daný shluk.
- *max:* Maximální hodnota pro daný shluk.
- *min:* Minimální hodnota pro daný shluk.
- *firstValue:* První nalezená hodnota pro daný shluk, dle řazení.
- *lastValue:* Poslední nalezená hodnot pro daný shluk, dle řazení.

Nastavení hodnot se může dle typu vizuálu lišit. U každého vizuálu však může být nastaven název (title), velikost, pozice v dashboardu a barevné zobrazení jednotlivých hodnot.

Interakce s vizuály

Ani interakce s vizuály se oproti ostatním analyzovaným aplikacím neliší. Jediná možnost navíc je zobrazení detailu hodnoty IRI sloupce, kdy může být vykonán

dotaz 6.3 pro získání více informací o vybraném záznamu a následně detail pomocí dialogu ihned zobrazen, což by u dat běžné povahy možné nebylo.

Filtrace

Jelikož může jeden dashboard obsahovat libovolný počet zdrojů dat, je vhodné nastavovat filtrace datových zdrojů dle stránky dashboardu, kdy se na různých stránkách mohou pohledy lišit. Dále může být filtr přidán i na konkrétní vizuál, kdy se jedná o ještě detailnější přiblížení v pohledu stránky.

Sdílení dashboardu

Podobně jako u ostatních zkoumaných aplikací je vhodné implementovat i možnost sdílení vytvořeného dashboardu. To je ulehčeno povahou webové aplikace, kdy stačí nastavení veřejně přístupného permanentního odkazu dashboardu pouze pro čtení.

6.5 Specifikace požadavků

Analýza nástrojů společně s navrženou metodou tvoří specifikaci požadavků na SW, které jsou zobrazeny v tabulce B.1 z přílohy B. Požadavky jsou rozděleny dle priority **P** na *kritické - A*, *možná rozšíření - B* (podstatné, ale metoda má přínos i bez nich) a *vhodná budoucí rozšíření - C* (užitečné, ale s nižší prioritou). Požadavky jsou členěny dle rozdělení procesů z části 4. Prioritizované jsou požadavky, které přinesou konkurenční výhodu oproti analyzovaným aplikacím, zejména funkcionalita podpory zpracování propojených dat. Dále je prioritizováno jádro funkcionality transformací a dashboardů.

Návrh aplikace zpracování a vizualizace dat

7

Pro implementování specifikovaných požadavků je nutné vytvoření netriviální aplikace. Jelikož se má jednat o webovou aplikaci s mnoha uživatelskými nastaveními, je nutná existence databáze pro ukládání aplikačních dat. Vizí je aplikace, kde jsou dotazy vykonávány serverovou částí aplikace, která následně transformuje a zpracovává získaná data. Data jsou poskytována webové aplikaci, která je používá v interaktivních dashboardech a pohledech pro další transformace.

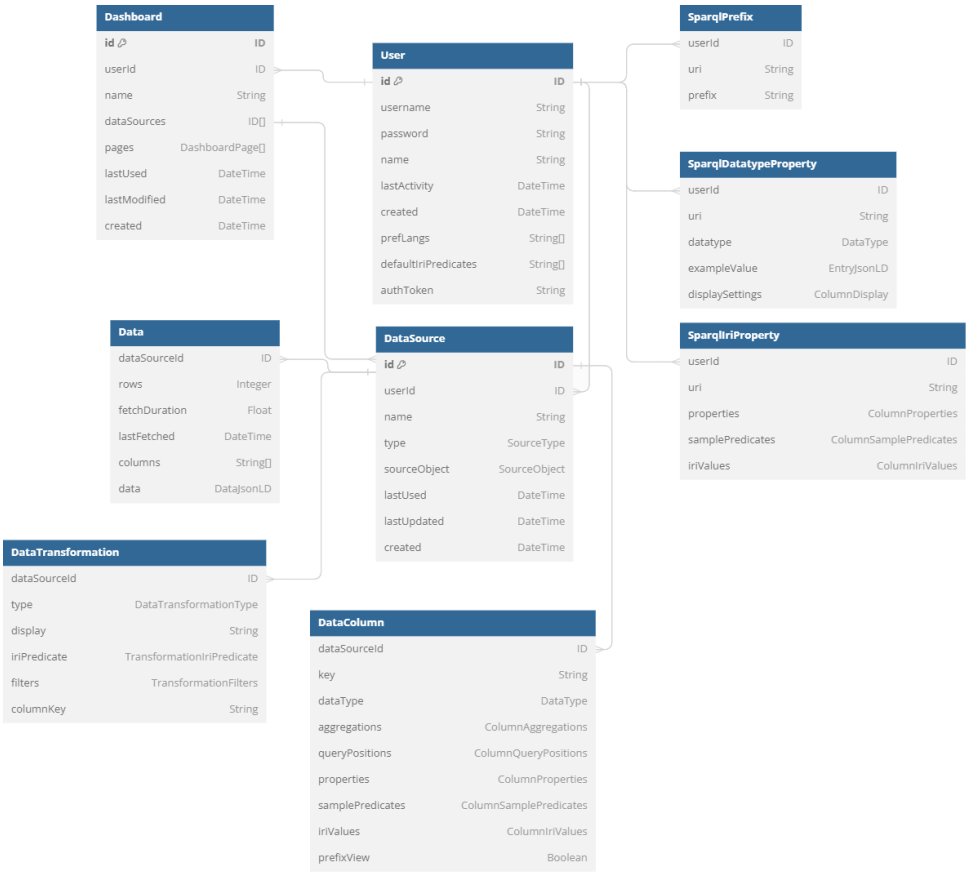
7.1 Databázový model

Na základě návrhu metod zpracování a vizualizace dat z předchozí kapitoly byl vytvořen obecný databázový model, který konkrétní podobu dat využívaných nebo modifikovaných těmito metodami definuje. Zjednodušený model je znázorněn diagramem na obrázku 7.1. Popsání jednotlivých entit úplného databázového modelu je pak věnována příloha C.

7.2 Volba technologií

7.2.1 RDF úložiště ontologií a slovníků

Volba konkrétní implementace RDF úložiště je zcela na uživateli. Zpravidla může aplikace k vlastnímu úložišti přistupovat dle nastavení stejným způsobem jako k externím SPARQL Endpointům. Vlastní RDF úložiště může v některých případech běžet i na stejném zařízení jako serverová část aplikace. V takovém případě se očekává kratší doba odezvy při komunikaci s vlastním úložištěm. Vhodné jsou například úložiště OpenLink Virtuoso [15] a Apache Jena Fuseki [16]. Možností je i použití vyhrazeného grafu na RDF úložišti. Nutná je možnost nahrávat nové slovníky, ontologie a možnost je modifikovat.



Obrázek 7.1: Zjednodušené schéma databázového modelu

7.2.2 Databáze

Pro implementaci databázového modelu aplikačních dat byla zvolena technologie Mongo DB. MongoDB je dokumentová databáze, která ukládá data ve flexibilních dokumentech typu JSON, což znamená, že pole se mohou v jednotlivých dokumentech lišit a struktura dat se může v průběhu času měnit. Vlastnosti databázové technologie: [32]

- Dotazy ad hoc, indexování a agregace v reálném čase poskytují výkonné způsoby přístupu k datům a jejich analýzy.
- MongoDB je ve svém jádru distribuovaná databáze, takže vysoká dostupnost, horizontální škálování a geografická distribuce jsou integrovány, snadno se používají.

- MongoDB je k dispozici zdarma.

Možností bylo i využití RDF úložiště jako databáze aplikačních dat. MongoDB bylo upřednostněno kvůli následujícím výhodám aplikačních dat oproti RDF databázi:

- Jednotný formát dat: SPARQL Endpoint umožňuje získání dat ve formátu JSON, což je přirozený formát pro MongoDB i pro webového klienta. Nebude nutná serializace do RDF a zpět do JSON.
- Komplexita modelu: Databázový model definuje komplexní vnořené objekty, obsažené jako atributy v jednotlivých entitách databáze. Implementace dynamických entit se složitými objektovými atributy je pro Mongo naprosto přirozená.

MongoDB bude použito pro aplikační data a RDF úložiště pro importované slovníky a ontologie. MongoDB umožní přímočařejší implementaci aplikační logiky, zatímco RDF úložiště poskytne strukturované úložiště pro ontologie a komplexnější dotazy. V případě vzniku aplikačních dat, jejichž podoba je přirozenější pro RDF úložiště, je vždy možnost s těmito daty pracovat v něm.

7.2.3 Server

Volba technologie na straně serveru žádá dobrou podporu zpracování dat a podporu práce se SPARQL dotazy a jejich daty. Dále je nutná podpora práce s Mongo DB. Cílem je volba přímočarých nástrojů, které umožní rychlý vývoj jednoduché aplikace.

Python

Tvůrci jazyku Python definují jako interpretovaný, objektově orientovaný, vysokoúrovňový programovací jazyk s dynamickou sémantikou. Jeho vysokoúrovňové vestavěné datové struktury, v kombinaci s dynamickým typováním a dynamickými vazbami, jej činí velmi atraktivním pro rychlý vývoj aplikací a pro použití jako skriptovací nebo spojovací jazyk propojující existující komponenty. Existence bohatých knihoven pro datovou analýzu z Pythonu dělá ideálního kandidáta pro zpracování dat. [33] Pro zpracování SPARQL dotazů existuje knihovna v Pythonu SPARQL Wrapper, která podporuje vykonávání SPARQL dotazů a základní práci s jejich výsledky. [34]

Flask

Flask je webový framework, modul jazyka Python, který umožňuje vývoj webových aplikací. Jeho součástí je malé a snadno rozšiřitelné jádro, neobsahuje ORM

(*Object Relational Manager*), ani podobné funkce. Má mnoho zajímavých funkcí, jako je směrování URL nebo šablonovací engine. [35]

Mongo Engine

MongoEngine je Document-Object Mapper (ORM, ale pro dokumentové databáze) pro práci s MongoDB v jazyce Python. Knihovna využívá jednoduché deklarativní API pro práci s databází nebo definici modelů. Knihovna implementuje serializace i deserializace JSON objektů, které odpovídají dokumentům nebo vnořeným dokumentům v databázi. [36]

Na straně serveru bylo ze zmíněných možností, které plně odpovídají cílům volby serverových nástrojů, rozhodnuto využít framework Flask běžící v jazyce Python.

7.2.4 Klient

Jelikož má být uživatelským rozhraním webová aplikace, je nutné využití jazyka JavaScript a jeho knihoven. Možností byli zejména tři základní JavaScriptové frameworky: Angular[1], Vue.js[2] a React.js[3]. Zejména z důvodu vlastní znalosti tohoto frameworku a obdobným možnostem jednotlivých frameworků ve využití potřebných knihoven v prostředí JavaScript, byl zvolen framework React.

React

React je JavaScriptová knihovna pro tvorbu uživatelských rozhraní. Je zaměřena na komponentovou architekturu, což znamená, že UI je rozděleno do samostatných komponent, které mohou být znovupoužity a dobře organizovány. Deklarativní přístup k vytváření uživatelského rozhraní umožňuje propojit data s vizualizačními komponentami. Pomocí principu hooků lze jednoduše docílit interaktivní povahy aplikace. [37]

Material UI

Material UI je open-source knihovna komponent React, která implementuje Material Design společnosti Google. Rozsah implementovaných komponent je komplexní a lze je ihned využívat pro tvorbu komplexnějších komponent. [38]

React Query

[1] Angular - https://angular.io/
[2] Vue.js - https://vuejs.org/
[3] React.js - https://react.dev/

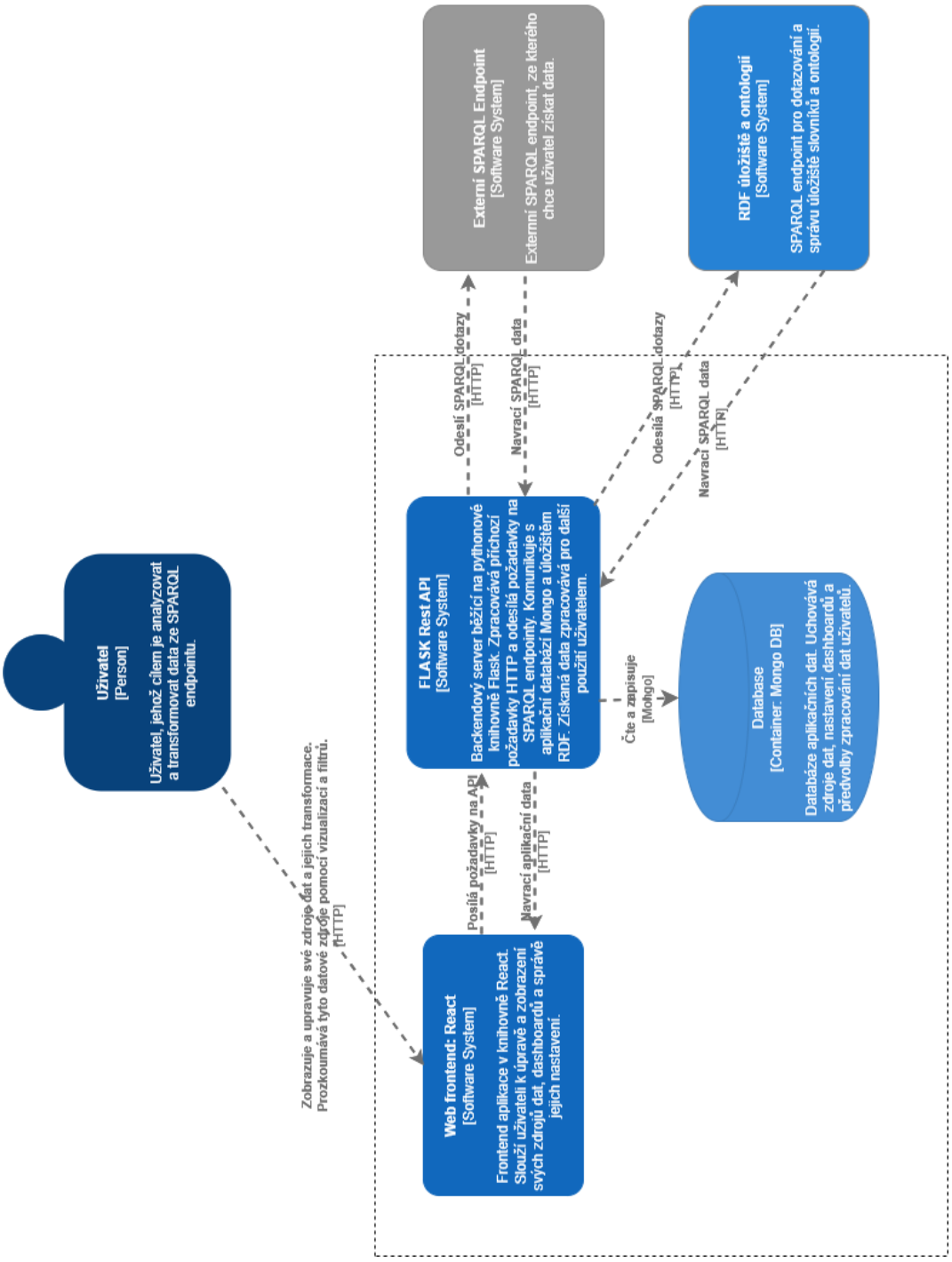
React Query je výkonná knihovna pro správu stavu aplikace React. Abstrahuje složitost načítání dat, ukládání do mezipaměti, synchronizace a umožňuje vývojáři soustředit se na implementaci uživatelského rozhraní. [39]

Chart.js

Chart.js je jedna z knihoven analyzovaných v části 5.8. Knihovna byla zvolena pro jednoduchost použití a značné možnosti rozšíření interaktivnosti.

7.2.5 Model nasazení

Popsaný technologický stack dává dohromady architekturu, která je vyjádřena modelem na obrázku 7.2. Aplikace serveru i backendu je zároveň dockerizována, což umožňuje jednodušší nasazení aplikace na odlišná zařízení.



Obrázek 7.2: Model nasazení aplikace

Implementace 8

8.1 Server

8.1.1 Struktura

Implementace serverové části je logicky rozdělena do několika modulů dle funkcionality, kterou dané moduly implementují. Aplikace vychází z MVC architektury, které odpovídá členění Model (models), View (views), Controller (services). Seznam souborů v kořenovém adresáři aplikace:

- **/shared:** Sdílený modul aplikace, který obsahuje zdroje pro základní nastavení aplikace: *app.py* (nastavení prefixu REST API a portu, sdílené proměnné s vlákny pro zpracovávání), *database.py* (nastavení připojení k aplikační databázi pomocí Mongo Engine), *api.py* (nastavení rozhraní REST API aplikace).
- **/models:** Implementuje ORM třídy všech modelů z aplikační databáze v jazyce Python. Modely odpovídají databázovému modelu aplikace, popsanému v příloze C.
- **/views:** Implementuje REST API aplikace a jeho obsluhu voláním služeb (services). Více v části 8.1.3.
- **/services:** Implementuje logiku aplikace a je rozdělen do následujících služeb: uživatelské služby (*UserService.py*), práce se soubory (zapisování, mazání a čtení, *FileProcessingService.py*), správa dashboardů (*DashboardService.py*), práce se SPARQL viz 8.1.4 (*RDFStorageService.py, SparqlOntologyService.py, SparqlService.py*), zpracování dat viz 8.1.5 (*DataSourceService.py, DataService.py, DataTransformationsService.py, DataColumnsService.py*).
- **/STORAGE:** Adresář STORAGE v kořenovém adresáři serveru slouží jako souborové úložiště. Uchovává soubory s výchozím SPARQL nastavením uživatelů (*/userDefaults* viz 8.1.7) a základními RDF slovníky pro import (*/vocabularies*) do případného vlastního úložiště. Funkcí adresáře je uchování dat

získaných ze SPARQL Endpointu nebo jiného zdroje. Data jsou uchovávána mimo databázi z důvodu optimalizace výkonnosti práce s nimi v podadresáři */data*.

- **/tests:** Modul implementuje UNIT testy, kterým je věnovaná část 9.1.
- *Soubory:* **main.py** (spuštění aplikace), **run_tests.py** (vstupní skript UNIT testů, sloužící pouze k spuštění testování), **requirements.txt** (obsahuje závislosti serverové aplikace, které jsou referencí na tento soubor v případě potřeby nainstalovány), **Dockerfile** (obsahuje instrukce pro konfiguraci a vytvoření Dockerového obrazu pro spuštění serverové části aplikace v kontejneru).

8.1.2 Uživatelské zdroje

V aplikaci nejsou aktuálně implementovány uživatelské role a pro práci s aplikací potřebuje uživatel účet (registrace je aktuálně povolena komukoliv). Není k dispozici žádný administrátorský panel a zdroj v aplikaci náleží právě jednomu uživateli. Jmenovitě může uživatel vlastnit a upravovat následující zdroje a nikdo jiný ke stejným zdrojům nemá přístup:

- *SPARQL nastavení*, existující pro uživatele maximálně jednou: předvolby prefixů, preferované jazyky, předvolby načítaných predikátů při zpracování IRI sloupců, předvolby nastavení, předvolby nastavení IRI sloupců dle predikátu.
- *Zdroje dat*, kterých uživatel může mít neomezený počet a každému zdroji náleží právě jedno z následujících nastavení: nastavení sloupců, nastavení transformací, data získaná ze zdroje dat.
- *Dashboardy*, kterých může uživatel mít neomezený počet a dashboard má: N libovolných zdrojů dat, které uživatel vlastní, libovolný počet stránek s libovolným počtem vizuálů (jeden vizuál zobrazuje jeden zdroj dat).

S uživatelským nastavením lze pracovat po autentizaci uživatele a u dashboardů a datových zdrojů je navíc nutné dané entity vlastnit.

8.1.3 REST API

Základní členění API volání odpovídá i rozdělení uživatelských zdrojů z předchozí části:

- `/user`: Přihlášení, registrace, ověření autentizace, správa.uživatelského účtu.
- `/dataSource`: Práce s datovými zdroji.

- `/data/<dataSourceId>`: Práce s daty, transformacemi a sloupci datového zdroje.
- `/dashboard`: Práce s dashboardy.
- `/sparql`: Práce se SPARQL nastaveními, získání metadat pro IRI hodnotu.

Pro ověření identity uživatele byl vytvořen dekorátor authenticate (UserView.py), kterým jsou obaleny všechny funkce API volání kromě přihlášení a registrace. Implementuje ověření pomocí tokenu, který musí odpovídat tokenu příslušícímu uživateli. Detailní dokumentaci REST API lze nalézt v příloze A.

8.1.4 Práce se SPARQL

`SparqlOntologyService.py` se zabývá pomocnými funkcemi pro práci se SPARQL. Implementuje funkce pro zpracování prefixů nebo proměnných a jejich pozic ze SPARQL dotazu. Obsahuje funkce pro generování SPARQL dotazů dle parametrů: seznam IRI, predikáty a preferované jazyky. Významná funkce je `getIriMeta`, která vykonává dotazy získávající vlastnosti IRI hodnot. Dotazy jsou vykonávany nad vzdáleným SPARQL Endpointem a nad vlastním úložištěm. Funkce výsledky spojí a vrátí všechna úspěšně získaná metadata.

`SparqlService.py` se zabývá vykonáním SPARQL dotazů (získání dat, nikoliv metadat) a zpracováním výsledků. Je implementována základní funkce `query`, která za pomoci knihovny `SPARQLWrapper` vykonává SPARQL dotazy a převádí je do zvoleného formátu. Jsou zde obsaženy funkce pro práci s uživatelskými SPARQL nastaveními a inicializace jejich výchozích hodnot, viz 8.1.7.

`RDFStorageService.py` obsahuje nastavení pro připojení k RDF úložišti a vykonávání dotazu. Je obsažen základ funkcí pro správu úložiště jako je import slovníků nebo zobrazení importovaných dat apod. Aktuálně je však nutné spravovat RDF úložiště slovníků a ontologií přímo tam, kde běží (např. Apache Jena Fuseki, která pro své úložiště má vlastní webové rozhraní).

8.1.5 Zpracování dat

`DataSourceService.py` implementuje funkce pro manipulaci s datovým zdrojem (smazání, úprava), validaci zdroje, kontrolu změny zdroje dat (nutnost nového dotazu) a získání metadat datového zdroje (například počet řádků z dat). Stěžejní je funkce `handleDataSourceChanged`, která spouští proces pro získání a zpracování nových dat z třídy `DataService`. Je volána při vytvoření zdroje nebo aktualizaci.

`DataService.py` je modulem pro zpracování dat. Implementuje čtení, zápis a mazání dat datového zdroje. Obsahuje funkce pro zpracování nově získaných dat, které spouští proces nového zpracování sloupců těchto dat z třídy

`DataColumnsService`. Jelikož má třída zodpovědnost za změny dat datového zdroje, spouští i transformace dat.

`DataColumnsService.py` implementuje práci se sloupci dat datového zdroje. Obsahuje funkce pro zpracování statistik sloupce, zpracování datového typu sloupce, získání a uložení jeho metadat v případě, že se jedná o IRI sloupec.

`DataTransformationsService.py` implementuje funkce pro správu transformací uložených k danému datovému zdroji. Aplikace těchto transformací se však nachází v DataService, kde se operativně tyto transformace spouští při aktualizaci zdroje nebo vynucením po úpravě transformací.

8.1.6 Paralelizace zpracování dat

Vykonávání dotazů, zpracovávání sloupců, ani transformace získaných dat, nemusí být vždy triviální a je nutná implementace paralelního zpracování, která umožní uživateli bez blokování ihned odpovědět a případně sdělit stav zpracování těchto dat. Ve zmíněném sdíleném modulu `shared/app.py` jsou definovány globální proměnné `queryingData`, `processingDataColumns` a `transformingData`, které požadavek implementují.

```
# data source QUERY
queryingData : dict[str, Thread] = {}
# data columns
processingDataColumns : dict[str, dict[str, Thread]] = {}
# data transformation working on DATA transforming
transformingData : dict[str, Thread] = {}
```

Zdrojový kód 8.1: Ukázka serverových proměnných pro paralelní zpracování dat

Proměnná `queryingData` uchovává procesy zpracovávání SPARQL dotazů. Po vytvoření nebo úpravě datového zdroje a spuštění procesu aktualizace nebo získání dat se vytvoří vlákno v slovníku `queryingData` s id datového zdroje. Uživatel je po celý průběh v případě potřeby informován o stavu zpracování dat, samotný proces svůj úspěšný nebo neúspěšný výsledek zapíše do databáze.

Proměnná `processingDataColumns` uchovává procesy zpracování sloupců. Po úspěšném získání dat je spuštěno zpracování sloupců, kdy každý sloupec je pro datový zdroj zpracováván ve vlastním vlákně. Pro IRI sloupce o mnoha rozdílných hodnotách zpracování zabírá mnohem více času než pro jednoduchý číselný sloupec. V případě změny dat již stačí pouze přepočítání statistik a získání případných chybějících metadat, která jsou nově pro data potřebná.

Proměnná `transformingData` uchovává procesy zpracování transformací. V případě, že existují nebo jsou vytvořeny nové transformace, jsou po znovuzískání a zpracování sloupců provedeny. Některé transformace z důvodu optimalizace vyžadují existenci již zpracovaných sloupců a procesy transformace běží až po jejich

zpracování. Některé transformace vynucují přepočítání statistik nebo nové zpracování sloupců. Mechanismus dle typu transformací určí, zda je potřeba přepočítat statistiky všech sloupců (například u odebrání řádků). V případě vytvoření nového sloupce jsou například jeho statistiky zpracovány přímo v transformaci.

8.1.7 Výchozí uživatelská nastavení aplikace

Pro každého nově registrovaného uživatele aplikace jsou zapsány výchozí SPARQL nastavení, která ovlivňují následné zpracovávání dat ze SPARQL dotazu. V následujícím textu bude představena podoba těchto výchozích hodnot a společně s tím struktura a funkcionalita těchto nastavení.

Uložené prefixy slouží k možnosti zobrazování IRI hodnot ve zkrácené prefixové podobě. Jejich inicializace probíhá pro základní všeobecné prefixy a dále prefixy z DBPedia, Wikidata a MRE.

Preferované jazyky jsou uloženy v uživatelském profilu a největší prioritu má první jazyk v jejich nastavení. V případě, že není možnost použití ani jednoho z prioritizovaných jazyků, jedná aplikace tak, jako by žádné priority jazyků neexistovaly, jinak by mohlo dojít ke ztrátě informace. V rámci výchozí inicializace je nastaven prioritně český jazyk a dále anglický.

Nastavení výchozích predikátů určuje načítaní vlastností pro jednotlivé IRI hodnoty sloupce. První nastavená hodnota je použita jako zobrazení daného sloupce. Příkladem je získání *rdfs:label* pro všechny hodnoty IRI sloupců.

Nastavení datových typů konfiguruje mapování tříd datových typů na datový typ v aplikaci. Rozpoznaný datový typ sloupce přiřadí sloupci definované nastavení. Výchozí nastavení obsahuje některé základní třídy datových typů ze slovníku XSD.

Konkrétní výchozí hodnoty je možné najít ve zdrojovém kódu aplikace v adresáři `STORAGE/userDefaults`.

8.2 Klient

8.2.1 Struktura

Struktura kořenového adresáře klienta je obdobná struktuře běžné Reactové aplikace: **/node_modules** (obsahuje všechny závislosti nainstalované pomocí NPM[1]), **/public** (obsahuje veřejné statické soubory jako jsou obrázky), **/src** (zdrojové kódy aplikace), **Dockerfile** (konfigurace a vytvoření Dockerového obrazu pro spuštění aplikace v kontejneru), **package.json** (obsahuje metadata o aplikaci a souhrn všech závislostí projektu), **tsconfig.json** (konfigurace nastavení kompilátoru jazyka TypeScript).

[1] NPM - Node Package Manager

Struktura adresáře *src*, která rozděluje moduly dle funkcionality:

- **/services/api:** Implementuje všechna API volání odpovídající API serverové části aplikace viz POPIS API pomocí knihovny AXIOS.
- **/services/utils:** Modul utils obsahuje pomocné funkce pro práci s daty napříč všemi obrazovkami aplikace:
  - *dashboard.ts:* Implementuje porovnání ekvivalence dashboardů nebo jejich stránek. Jsou zde funkce pro získání výchozího objektu dashboardu a dashboardové stránky, funkce pro práci s dashboardy, porovnání ekvivalence dashboardů nebo jejich stránek. Je zde obsaženo získání výchozího objektu dashboardu a dashboardové stránky.
  - *dataSource.ts:* Implementuje funkce pro práci se zdroji dat, porovnání ekvivalence datových zdrojů a získání výchozího objektu zdroje dat.
  - *data.ts:* Implementuje funkce pro práci s hodnotami dat. Je zde funkce porovnání libovolných hodnot pro řazení a dále funkce pro převedení hodnot do podoby dle nastavení jejich vykreslení (například zobrazení datumu, čísla).
  - *sparql.ts:* Implementuje funkce pro práci s IRI hodnotami převedení do prefixové notace nebo získání textové podoby v případě vykreslení IRI s alternativní hodnotou.
  - *table.ts:* Implementuje funkce pro kopírování objektů nebo porovnání ekvivalence objektů.
  - *visual.ts:* Obsahuje výpočetní funkci `getVisualChartData` předzpracování dat pro vizuál. Funkce dle nastavení vypočítá konkrétní hodnoty pro osy X a Y. Zpracovaná data jsou v podobě pro vizuály knihovny ChartJS.
- **/types:** Obsahuje definice datových typů aplikace, které odpovídají modelům na serverové části aplikace a modely pro obsluhu komponent.
- **/components:** Obsahuje dílčí části komponent, ze kterých jsou složeny rodičovské komponenty stránek viz 8.2.2.
- **/pages:** Obsahuje hlavní komponenty, které představují stránky aplikace viz 8.2.3.
- *Soubory:* **App.tsx** (vstupním bod aplikace, definice URL směrování), **constants.ts** (globální konstanty, adresa serverového API), **images.ts** (cesty k obrázkům), **index.tsx** (inicializace aplikace).

8.2.2 Přehled komponent

Komponenty aplikace jsou rozděleny dle funkcionality do modulů. Modul */user* obsahuje pouze komponentu poskytující uživatelský stav aplikace.

Modul */sparql* implementuje všechny komponenty SPARQL nastavení uživatele. Dále je obsažena důležitá komponenta `RenderSparqlValue.tsx`, která vykresluje dle nastavení RDF data aplikace.

Modul */dataSources* implementuje komponenty, které dohromady umožňují funkcionalitu obrazovky datových zdrojů včetně tabulek zobrazení dat, nebo obsažené komponenty nastavení zdroje dat (SPARQL nebo soubor).

Modul */dashboard* implementuje komponenty pro obrazovky dashboardů, komponentu `AddDataSourceMenu.tsx` (menu přidání zdroje dat do dashboardu) a `AddGraphMenu.tsx` (menu přidání grafu do dashboardu). `RenderVisual.tsx` je pak komponenta pro vykreslení vizuálu pomocí knihovny ChartJS.

Modul */dataRender* obsahuje implementaci dynamických formulářů `ColumnsRender.tsx`, který vykresluje skupinu proměnných dle datového typu a spravuje změnu jejich stavu. V podadresáři */columns* lze nalézt vykreslovací komponenty jednotlivých datových typů formulářových hodnot (`PasswordRender.tsx`, `TextAreaRender.tsx`, `TextRender.tsx`, `DateTimeRender.tsx`, `FileInputRender.tsx`, *OptionsRender.tsx*).

Modul */columnSettings* obsahuje komponenty všech menu a podmenu, které jsou používané v rámci aplikace pro práci se sloupci. */fileUpload* implementuje komponentu zobrazení formuláře pro nahrání souboru, případně zobrazení nahraného souboru. Tato komponenta je pak využívána pro `FileInputRender.tsx`.

Modul */layout* implementuje navigační menu aplikace, komponentu zobrazující stav načítání nebo informační lištu, která zobrazuje chyby nebo odezvy komunikace se serverem. Modul */dialog* implementuje konfirmační dialog a komponentu pro dynamické vyskakovací komponenty.

Modul */modals* implementuje komponentu `DataModal.tsx`, která slouží pro vyskakovací dynamické formuláře. */buttons* obsahuje komponentu kopírovacího tlačítka `CopyToClipboardButton.tsx`, které zkopíruje nastavený text do schránky uživatele. */icons* implementuje komponentu `DatatypeIcons.tsx`, která vykresluje zobrazení datového typu.

8.2.3 Obrazovky aplikace

Je implementováno celkem 11 obrazovek, jejichž používání i s grafickými ukázkami je zdokumentováno v příloze G.

HomePage.tsx je úvodní obrazovkou dostupnou na URL `/` a obsahuje stručný popis aplikace. Na URL `/login` je dostupná obrazovka pro přihlášení *LoginPage.tsx*. Registraci přísluší URL `/register` a komponenta *RegisterPage.tsx*. Stránka

NotFoundPage.tsx je zobrazena v případě, že se uživatel pokouší přistoupit na URL, které neexistuje, a *UnauthorizedPage.tsx*, v případě, že se uživatel pokouší přistoupit na zdroj, na který nemá přístupová práva.

SparqlSettingsPage.tsx představuje stránku SPARQL nastavení uživatele. Lze přepínat mezi nastavením zobrazení datových typů, výchozích IRI predikátů, prefixů a preferovaných jazyků. Danému nastavení pak odpovídá URL `/sparq-settings/<typ-nastavení>`.

DataSourcesListPage.tsx je stránkou aplikace, kde je zobrazen seznam všech datových zdrojů uživatele. Uživatel odsud může zdroje smazat, kopírovat, aktualizovat, nebo se přesměrovat na konkrétní zdroj, či vytvoření nového datového zdroje. Odpovídajícím URL je `/data-sources`. *DataSourcePage.tsx* je zobrazením detailu konkrétního datového zdroje. V obrazovce může uživatel přepínat mezi nastavením zdroje (SPARQL dotaz, url, údaje nebo soubor), zobrazením dat (tabulka s daty přes celou obrazovku) a transformacemi (boční a horní panel s transformacemi).
V případě, že je hodnota *id* v URL *new-source* jedná se o obrazovku vytvoření nového zdroje, která je totožná, až na fakt, že jsou dostupná jen nastavení zdroje, nikoliv data a transformace. Detail zdroje je dostupný na URL `/data-sources/<id>`.

DashboardListPage.tsx je obrazovkou na URL `/dashboards`, kde je zobrazen seznam všech dashboardů uživatele. Uživatel může dashboardy smazat nebo kopírovat. Dále je možné přesměrování na dashboard nebo vytvoření nového dashboardu. *DashboardPage.tsx* je zobrazením detailu konkrétního dashboardu na URL `/dashboard/<id>`. V obrazovce může uživatel přepínat mezi stránkami dashboardu, nebo je upravovat. Obdobně jako u datových zdrojů se v případě, že je hodnota *id* v URL *new-dashboard*, jedná se o obrazovku vytvoření nového dashboardu. Po pravé straně obrazovky se nachází menu pro přidání a úpravu datových zdrojů a v případě výběru konkrétního vizuálu je v pravém dolním rohu i nastavení daného vizuálu. Horní panel poskytuje možnosti operací nad dashboardem, kde se aktuálně nachází jen možnost přidání grafu.

SparqlIriMetaPage.tsx je obrazovkou, která se otevírá jako nové okno a je dostupná na URL `/sparql/iri-meta/<endpoint>/<iri>`. Slouží k implementaci *details on demand* průzkumu IRI subjektů. Jejím obsahem je tabulka se všemi získanými vlastnostmi daného subjektu. Čarou jsou odděleny uživatelem přednastavené vlastnosti a ostatní vlastnosti tohoto IRI.

8.2.4 Zobrazení literálů

Nastavení každého sloupce literálů může proběhnout již před dotazováním, nebo až manuálně po získání dat, pokud například datový typ nebyl rozpoznán. Pro všechny implementované datové typy lze data řadit, nebo filtrovat jejich hodnoty. Sloupce

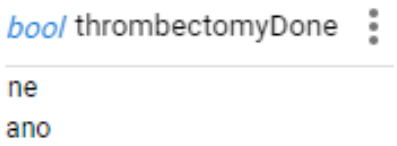
lze duplikovat a například každý zobrazit jinak. Lze nastavit jejich zarovnání, šířku nebo název.

Boolean

Na obrázku 8.1 je možné vidět výchozí zobrazení booleovského sloupce z dotazu D.2 ve Virtuoso. Obrázek 8.2 znázorňuje možnost zobrazení takového sloupce v implementované aplikaci. Pro zobrazení hodnot lze nastavit libovolné řetězce, kdy na obrázku je nastaveno *ano* pro pravdu a *ne* pro nepravdu.

stroke_thrombectomyDone_stroke_t_baseline
"false"^^<http://www.w3.org/2001/XMLSchema#boolean>
"true"^^<http://www.w3.org/2001/XMLSchema#boolean>

Obrázek 8.1: Ukázka klasického zobrazení boolean



Obrázek 8.2: Ukázka implementovaného zobrazení boolean

Číslo

Na obrázku 8.3 je znázorněno výchozí zobrazení číselného sloupce v aplikaci *dbpedia/snorql* z D.3 - Populace měst. Obrázek 8.4 pak znázorňuje možnost zobrazení takového sloupce v implementované aplikaci. Číselné hodnoty jsou zarovnány doprava, je nastaveno seskupování čísel po třech znacích. Je k dispozici i možnost nastavení počtu číslic za desetinnou čarou a zobrazený oddělovač celočíselné části a desetinné části.

Datum a čas

Na obrázku 8.5 je možné vidět výchozí zobrazení sloupce datumu D.2 - Akutní intervence v aplikaci Virtuoso mre.zcu.cz. Obrázek 8.6 z aplikace zobrazení formátuje datum tak, aby byl zobrazen den, textová podoba měsíce, rok a hodina. Řazení a seskupování následně pracuje dle zobrazovaných hodnot (například je zobrazen měsíc, je řazen jen měsíc).

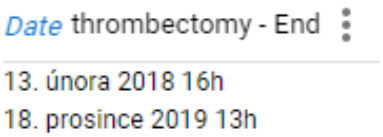
population
56864
127328

Obrázek 8.3: Ukázka klasického zobrazení čísla

Obrázek 8.4: Ukázka implementovaného zobrazení čísla

stroke_thrombectomyEndDateTime_stroke_t_baseline
"2018-02-13T16:32:00+0100"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"2019-12-18T13:40:00+0100"^^<http://www.w3.org/2001/XMLSchema#dateTime>

Obrázek 8.5: Ukázka klasického zobrazení datumu



Obrázek 8.6: Ukázka implementovaného zobrazení datumu

8.2.5 Zobrazení IRI

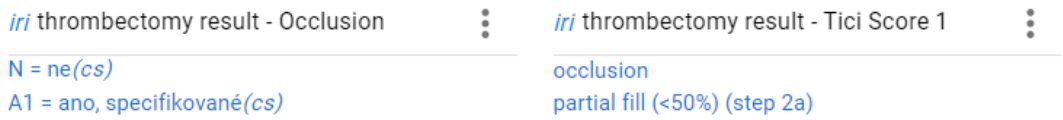
Na obrázku 8.7 je možné vidět výchozí zobrazení sloupců IRI D.2. Po předzpracování a výchozím načtení hodnoty label, však jsou zobrazeny hodnoty v lidsky čitelné podobě, jak znázorňuje obrázek 8.8.

IRI hodnoty mohou být zobrazeny načtenou vlastností. Může nastat případ, kdy není žádná z vlastností nalezena. Pak je v aplikaci možné alespoň zobrazit hodnoty v prefixové podobě, jak je znázorněno na obrázku 8.9, kde jsou rovněž hodnoty zarovnány na střed.

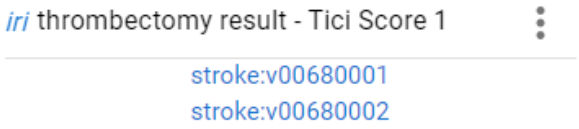
Zobrazení hodnot IRI v aplikaci z obrázku 8.7 umožňuje na IRI hodnoty klik-

stroke_thrombectomyResultOcclusion_stroke_t_baseline	stroke_thrombectomyResultTICIScore1_stroke_t_baseline
https://mre.zcu.cz/ontology/stroke.owl#v00670001	https://mre.zcu.cz/ontology/stroke.owl#v00680001
https://mre.zcu.cz/ontology/stroke.owl#v00670003	https://mre.zcu.cz/ontology/stroke.owl#v00680002

Obrázek 8.7: Ukázka klasického zobrazení IRI číselníků



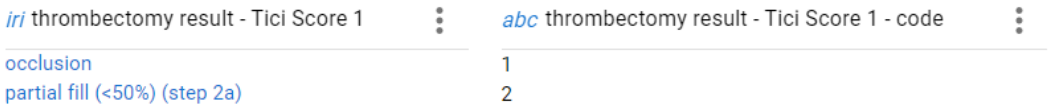
Obrázek 8.8: Ukázka implementovaného zobrazení IRI číselníků



Obrázek 8.9: Ukázka implementovaného prefixového zobrazení IRI

nout. Po kliknutí nastane stahování celého souboru ontologie. Implementovaná aplikace po kliknutí například na IRI hodnotu *stroke:v00680001* z obrázku 8.9 otevře nové okno, které představuje detail IRI položky s načtenými všemi nastavenými predikáty, a pod čarou další vlastnosti subjektu. Detail IRI hodnoty je znázorněn na obrázku F.4.

V případě, že se vlastnost subjektu zdá uživateli užitečná, může ji několika kliknutími přidat do celého IRI sloupce jako variantu zobrazení, nebo z dané vlastnosti vytvořit celý nový sloupec. Ukázka přidaného sloupce *form:code* na obrázku 8.10.



Obrázek 8.10: Ukázka přidaného sloupce *form:code*

8.2.6 Dashboardy

Tvorba dashboardů a vizualizací vychází z funkcionalit zmíněných u zobrazení sloupců. Všechny aktuálně podporované vizuály v aplikaci nastavují seskupovací osu X a následně osu Y pro konkrétní agregace hodnot. Ukázka celého dashboardu v aplikaci je na obrázku F.5, který používá data z dotazu D.3 - populace zemí a měst. Lze vidět, že IRI sloupce států jsou zobrazeny textovou podobou. Po průzkumu dat (overview first) je aplikována filtrace, která ponechává jen 4 státy (zoom and filter), jejichž celková populace je reprezentována na levém vizuálu s nadpisem *Populace států*. Tažení myši na vizuál zobrazuje detail konkrétní zobrazené hodnoty (details on demand). Po pravé straně je pak radarový vizuál, který porovnává populaci největších měst daného státu a jejich celkovou populaci. Vizuál používající nezpracované IRI hodnoty je vyobrazen na obrázku F.2 z dotazu D.2. Porovnání s vizuálem, který zobrazuje čitelnou podobu tohoto IRI číselníku na obrázku F.3, jasně dává najevo relevanci takového předzpracování.

Poslední ukázka na obrázku F.6 obsahuje vizuál, kde je nastaveno seskupení dle datumu (který je zobrazen pouze jako měsíc), a boolean hodnoty indikující, zda byla provedena trombolýza. Pro tyto seskupující proměnné je zobrazen celkový počet případů mrtvice za daný měsíc (modrá) a maximální počet provedených pasáží odstranění trombektomie v daném měsíci (červená).

Testování 9

V průběhu celé implementace práce byly implementované komponenty důkladně testovány. Práce byla průběžně pomocí zmíněných dockerových containerů nasazována na virtuální privátní server Západočeské univerzity (Open Nebula), kde byla stále aktuální implementace testována i vedoucím práce. Na serverové části byly implementovány automatické jednotkové testy. Manuálním testováním přes webové rozhraní aplikace je testováno, zda je uživatel o stavech aplikace informován a že tyto stavy odpovídají očekávanému scénáři.

9.1 Jednotkové testy

Jednotkové testy jsou na serverové části k nalezení v adresáři `/tests` a jejich vstupním skriptem je `run_tests.py` v kořenovém adresáři aplikace. Testy jsou implementované pomocí Pythonové knihovny *pytest*, kterou autoři popisují jako framework usnadňující psaní malých, čitelných testů, který dokáže podporovat komplexní funkční testování aplikací a knihoven. [40] V rámci vstupního skriptu je nastavena detailnost výpisu jednotlivých testů a předán adresář se samotným testováním, ze kterého si již knihovna všechny testovací moduly, jejich nastavení a testy, spustí. Knihovna očekává mírně odlišné názvy souborů a funkcí, než jsou používány v aplikaci.

Testovací data

Testování předpokládá využití endpointu s předem známým obsahem. Do úložiště je nutné nahrát testovací data, která jsou obsažena v příloze A. Testovací data obsahují data filmů z IMDB a byla získána z repozitáře *Programming Stardog: Examples*[1]. Ukázkový záznam z testovacího datasetu:

```
t:tt5929776
  :actor n:nm0000734 , n:nm7051652 , n:nm3377159 ;
  :description "A look at how climate change affects..." ;
  rdfs:label "Before the Flood" ;
```

[1]Programming Stardog: Examples - https://github.com/stardog-union/stardog-examples

```
5   :author n:nm0598531 ;
6   :director n:nm0001770 ;
7   :genre "Documentary" , "News" ;
8   :contentRating "PG" ;
9   :copyrightYear 2016 ;
10  :rating "8.4"^^xsd:float ;
11  :keyword "pope" , "environment" , "vatican", <...>  ;
12  :language "English" ;
13  :storyline "A look at how climate change affects... " .
```

Zdrojový kód 9.1: Ukázkový záznam testovacích dat

Vlastnosti filmů a dotazy, které jejich dataset umožňuje, jsou, pro své komplexní možnosti automatického testování a dostatečný rozsah (*193 370 trojic, 8 MB*), vhodné. Popsané testování proběhlo nad lokálním úložištěm běžícím na službě Apache Jena Fuseki. Nastavení připojení k testovacímu SPARQL Endpointu je možné v souboru **conftest.py**. Soubor obsahuje funkce, které poskytují testovacím modulům uživatele a zdroj dat, v případě neexistence jsou těmito funkcemi vytvořeny.

Testování uživatelských služeb

Modul **test_user_service.py** implementuje jednotkové testy, které se zabývají ověřením funkcionality služeb práce s uživateli: `test_user_authorization` - ověřuje funkcionalitu autentizace uživatele, zašifrování hesla nebo získání uživatele podle autentizačního tokenu a přihlašovacího jména, `test_create_user` - ověřuje funkcionalitu vytvoření uživatele a příslušných SPARQL nastavení.

Testování SPARQL služeb

Modul **test_sparql_service.py** implementuje testování SPARQL služeb aplikace: `test_parse_query_positions` - ověřuje zpracování pozic z proměnných SPARQL dotazu, `test_user_RDF_storage` - testuje komunikaci s RDF úložištěm ontologií dle aplikačních nastavení.

Testování služeb pro práci se zdroji dat

Modul **test_data_source_service.py** implementuje testy pro práci se zdroji dat, včetně jejich zpracování: `test_source_changed` - ověřuje rozpoznání nutnosti opětovného dotazu při změně nastavení zdroje dat,
`test_create_invalid_data_source` - testuje chybné vytvoření zdroje s neexistujícím endpointem nebo nesprávným nastavením, je ověřen vznik odpovídající chybové hlášky, `test_create_valid_data_source` - ověřuje celou metodu automatického zpracování dat, netriviální dotaz je vykonán nad testovacím endpointem a je testováno správné získání dat a zpracování sloupců včetně načtených IRI vlastností a získaných nastavení sloupce dle datových typů.

Testování transformačních služeb

Modul **test_transformations_service.py** testuje funkcionalitu transformačních metod: `test_remove_column` - kontroluje správnou funkcionalitu odebrání sloupce včetně smazání z dat i z databáze, `test_copy_column` - testuje transformace kopírování sloupce, včetně kopírování metadat, vytvoření nového sloupce v databázi i v datech. `test_filter` - testuje filtrování dat se správným přepočítáním statistik všech sloupců a odebrání odpovídajících řádků dle filtrovacích podmínek, `test_add_iri_column` - testuje transformaci přidání sloupce, vycházejícího z IRI hodnoty jiného sloupce, ověřuje správně získané vlastnosti nového sloupce včetně statistik a správný výskyt hodnot dle výskytu IRI záznamů původního sloupce.

9.2 Manuální testování

Manuální testování aplikace probíhalo přímo přes webového klienta aplikace, testuje také správnou integraci s běžícím aplikačním serverem. Jsou testovány scénáře pokrývající všechny obrazovky aplikace a interakci s nimi, scénáře jsou shlukovány dle funkcionality. Je uveden popis testovaného požadavku, scénář testování a PASS nebo FAIL, dle výsledku testu.

9.2.1 Uživatel

Nepřihlášený uživatel (NU) - **NU1**: *přístup pouze do registrační, přihlašovací a domovské obrazovky*. Přepnutí na jinou stránku i pomocí přímého URL uživatele přesměrovává na obrazovku přihlášení. `PASS`. **NU2**: *ošetření neexistujícího URL*. Přístup na neexistující stránku pomocí přímého URL uživatele přesměrovává na obrazovku s informací o neexistenci stránky. `PASS`.

Přihlášení (P) - **P1**: *zakázané odeslání přihlašovacího formuláře s nevalidními údaji*. Uživatel nezadá přihlašovací jméno ani heslo, požadavek na přihlášení se odešle a aplikace zobrazí zprávu *BAD REQUEST*. `FAIL`. **P2**: *nepovolené přihlášení se špatnými údaji*. Uživatel validně vyplňuje nesprávné údaje a odesílá formulář. Je zobrazena chybová zpráva *No such user or bad credentials* `PASS`. **P3**: *úspěšné přihlášení se správnými údaji a přesměrování na původní stránku*. Uživatel kliká na obrazovku zdrojů dat a je přesměrován do přihlášení. Uživatel správně vyplňuje existující údaje a odesílá formulář. Je přesměrován na obrazovku zdrojů dat a v pravém horním menu je zobrazeno jeho jméno. `PASS`.

Registrace (R) - **R1**: *zakázané odeslání přihlašovacího formuláře s nevalidními údaji*. Uživatel nezadá přihlašovací jméno ani heslo, požadavek o přihlášení se neodešle a aplikace zobrazí zprávu *Login and password are required*. `PASS`.
R2: *nepovolená registrace s existujícím přihlašovacím jménem*. Uživatel zadá

v registraci přihlašovací jméno *dev*. Odesílá formulář a je zobrazena chybová zpráva *User already exists* `PASS`. **R3**: *úspěšná registrace a vytvořené výchozí nastavení*. Uživatel zadává údaje *test2* s heslem *test* a odesílá formulář. Je přesměrován na obrazovku zdrojů dat a v pravém horním menu je zobrazeno jméno *test2*. Přepíná do obrazovky zdrojů, kde je existující nastavení datových typů, prefixů, predikátů i jazyků. `PASS`.

9.2.2 SPARQL nastavení

Nastavení jazyka (NJ) - **NJ1**: *uložené přidání a odebrání jazyka*. Do seznamu jazyků *cs, en* je přidán záznam s jazykem *de*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. Po aktualizaci obrazovky je mezi jazyky stále záznam *de*. Záznam je odstraněn a je aktualizována obrazovka. Záznam mezi jazyky není. `PASS`. **NJ2**: *nepovolené přidání existujícího jazyka*. Do seznamu jazyků *cs, en* je přidán záznam s jazykem *cs*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. Již existující jazyk by neměl být uložen. `FAIL`.

Nastavení prefixu (NPF) - **NPF1**: *uložené přidání a odebrání prefixu*. Do seznamu prefixů je přidán záznam s prefixem *foaf: <http://xmlns.com/foaf/0.1/>*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. Po aktualizaci obrazovky je mezi prefixy záznam zobrazen. Záznam je odstraněn a je aktualizována obrazovka. Záznam mezi prefixy není. `PASS`. **NPF2**: *nepovolené přidání existujícího prefixu*. Do seznamu prefixů je přidán záznam s již obsaženým URI prefixu *https://www.w3.org/2001/XMLSchema#*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. `FAIL`. **NPF3**: *zakázané přidání nevalidního prefixu*. Do seznamu prefixů je přidán záznam s URI prefixu bez # nebo / na svém konci a nevalidní hodnotou *nonsense-uri*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. `FAIL`.

Nastavení predikátů (NPD) - **NPD1**: *uložené přidání a odebrání predikátu*. Do seznamu predikátů je přidán záznam s predikátem *rdf:type*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. Po aktualizaci obrazovky je mezi predikáty záznam zobrazen. Záznam je odstraněn a je aktualizována obrazovka. Záznam mezi predikáty není. `PASS`. **NPD2**: *nepovolené přidání existujícího predikátu*. Do seznamu predikátů je přidán záznam s již obsaženým URI predikátu *rdfs:label*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. `FAIL`. **NPD3**: *zakázané přidání nevalidního predikátu*. Do seznamu predikátů je přidán záznam s nevalidní

hodnotou *nonsense-uri*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. FAIL.

Nastavení datových typů (ND) - **ND1**: *uložené přidání a odebrání nastavení*. Do nastavení datových typů je přidán záznam s URI *xsd:nonNegativeInteger* s datovým typem number a zarovnáním doprava. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. Po aktualizaci obrazovky je mezi mapováními záznam zobrazen. Záznam je odstraněn a je aktualizována obrazovka. Záznam mezi hodnotami není. PASS. **ND2**: *nepovolené přidání existujícího predikátu*. Do nastavení je přidán záznam s již obsaženým URI datového typu *xsd:int*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. FAIL. **ND3**: *zakázané přidání nevalidního predikátu*. Do nastavení je přidáno nastavení datového typu s nevalidní hodnotou *nonsense-uri*. Je stisknuto tlačítko uložení a zobrazena informace o úspěšném uložení nastavení. FAIL.

9.2.3 Zdroj dat

Seznam zdrojů dat (SZD) - **SZD1** - *úspěšné zkopírování zdroje dat*. VZD4 a v seznamu zdrojů dat je stisknuto tlačítko kopírování. Tabulka po chvíli zobrazí zkopírovaný zdroj na začátku seznamu. Po otevření odpovídají data i sloupce původnímu zdroji dat. PASS. **SZD2**: *úspěšné smazání zdroje dat*. VZD4 a v seznamu zdrojů dat je stisknuto tlačítko odstranění zdroje a dialog je potvrzen. Tabulka se po chvíli aktualizuje a smazaný zdroj se v seznamu nenachází. PASS. **SZD3**: *úspěšná aktualizace zdroje dat*. VZD4 a v seznamu zdrojů dat je stisknuto tlačítko aktualizace zdroje. U zdroje je zobrazena animace načítání a po chvíli zmizí. Zdroj dat obsahuje aktualizovaná dat.

Vytvoření zdroje dat (VZD) - **VZD1**: *zákaz prázdného názvu zdroje dat*. Je nastaven zdroj dat s dotazem D.3. Výchozí název je smazán a ponecháno prázdné pole. Tlačítko vytvoření zdroje je skryto. PASS. **VZD2**: *zákaz neplatného URL*. Je nastaven zdroj dat s dotazem D.3. Je nastaveno URL *nonsense-url* a stisknuto tlačítko vytvoření zdroje. Je zobrazena chyba *Invalid URL provided*. PASS. **VZD3**: *zákaz neplatného SPARQL*. Je nastaven zdroj dat s dotazem *nonsense-sparql* a stisknuto tlačítko vytvoření zdroje. Je zobrazena chyba *Invalid sparql query provided*. PASS. **VZD4**: *úspěšné vytvoření zdroje*. Je nastaven zdroj dat s dotazem D.3, URL *https://dbpedia.org/sparql* a stisknuto tlačítko vytvoření zdroje. Uživatel je přesměrován na detail vytvořeného zdroje dat a po chvíli načítání jsou zobrazena odpovídající data. PASS.

Úprava dat (UD) - **UD1**: *úspěšné smazání sloupce*. VZD4 a na sloupci *country* je otevřeno menu nastavení. Stisknutí smazání sloupce vyžádá nejprve v dia-

logovém okně konfirmaci a následně se obrazovka znovu načte a sloupec se v datech již nenachází. PASS. **UD2**: *úspěšné kopírování sloupce*. VZD4 a na sloupci *country* je otevřeno menu nastavení. Stisknutím tlačítka kopírovat je obrazovka znovu načtena a na konci se nachází nový sloupec se suffixem *_copy*. PASS. **VZD3**: *úspěšná filtrace*. VZD4 a na sloupci *country* je otevřeno menu nastavení. Ve spodním menu je zvoleno třídění dle IRI hodnot a označena pouze hodnota *India*. Je stisknuta aplikace filtrů a v tabulce jsou zobrazeny jen řádky obsahující hodnoty *country* India. PASS

9.2.4 Dashboard

Seznam dashboardů (SD) - **SD1** - *úspěšné zkopírování dashboardu*. VD2 a v seznamu dashboardů je stisknuto tlačítko kopírování. Obrazovka po chvíli aktualizuje tabulku a zkopírovaný dashboard je na začátku seznamu. Dashboard odpovídá původnímu zobrazení. PASS. **SD2**: *úspěšné smazání dashboardu*. VZD4, v seznamu dashboardů je stisknuto tlačítko odstranění dashboardu a dialog je potvrzen. Obrazovka po chvíli aktualizuje tabulku a smazaný dashboard se v seznamu nenachází. PASS.

Vytvoření dashboardu (VD) - **VD1**: *zákaz prázdného názvu dashboardu*. V seznamu dashboardů je stisknuto tlačítko vytvoření nového dashboardu a uživatel je přesměrován. Výchozí název je smazán a ponecháno prázdné pole. Tlačítko vytvoření dashbaordu je stisknuto a dashboard je úspěšně uložen. FAIL. **VD2**: *úspěšné vytvoření dashboardu*. Je nastaven název dashboardu *test* a přidán zdroj dat z VZD4. Je stisknuto tlačítko vytvoření dashboardu. Uživatel je přesměrován na detail vytvořeného dashboardu. PASS.

Vizuály (V) - **V1**: *úspěšné přidání vizuálu*. VD2 a je stisknuto tlačítko přidání grafu. Je zvolen sloupcový graf a záhy se objeví v levém horním kraji dashboardu. Je mu přiřazeno nastavení osy X: sloupec *country*, a osy Y: sloupec *population*. Graf zadané hodnoty vykresluje. PASS.

Celkem bylo vykonáno 32 manuálních testů, z nich 23 prošlo a 9 selhalo. Neúspěšné testy zahrnují zejména neověřené kontroly polí formulářů a jejich uložení. Veškeré jádro funkcionality však testováním prošlo.

Validace 10

Na validačních dotazech, jejichž popis je obsažen v příloze D), bude ověřena výkonnost zpracování jednotlivých sloupců a dotazů. Bude také zvalidována implementovaná funkcionalita dle specifikace požadavků.

10.1 Výkonnost automatického zpracování

Klientská i serverová část aplikace při provádění metrik běžela mimo dockerové containery přímo na fyzickém zařízení. Výkonnostní metriky automatické metody zpracování byly měřeny na zařízení se specifikacemi: *procesor* Intel(r) Core(™) i7-7700HQ CPU @ 2.80GHz, jádra: 4, logické procesory: 8; *fyzická paměť* 8 GB; *operační systém* Microsoft Windows 10 Home x64; *odezva internetového připojení* 22 ms; *rychlost stahování* 6.60 Mbit/s; *rychlost nahrávání* 3.54 Mbit/s; Python *3.11.4*, Node.js *18.17.1*.

10.1.1 Vykonání SPARQL dotazu

Trvání SPARQL dotazu, ani jeho průběh, aplikace neovlivní a je závislý zcela na dotazovaném SPARQL Endpointu. Pro úplnost je přesto uvedena tabulka 10.1 s informacemi o vykonání validačních dotazů. Průměrný čas vykonání SPARQL dotazu byl měřen v pěti po sobě jdoucích pokusech bez restartu zařízení.

Tabulka 10.1: Statistiky vykonání validačních dotazů

Dotaz	Průměrný čas	Řádky	Sloupce	Endpoint
Dotaz D.1	6s	5 882	14	mre.zcu.cz
Dotaz D.2	8s	5 819	22	mre.zcu.cz
Dotaz D.3	25s	10 000	3	dbpedia.org
Dotaz D.4	230s	129 367	6	mre.zcu.cz

10.1.2 Zpracování statistik sloupce

Průměrné časy zpracování statistik jednotlivých sloupců v tabulce 10.2 jsou zkoumány na pěti po sobě jdoucích měřeních zpracování dat bez restartu zařízení. Nejsou uvedeny hodnoty všech sloupců, pouze základní vzorek od každého datového typu a distribuce hodnot, která rolišuje počet různých hodnot a velikost datové sady. Zpracování probíhá paralelně a je měřena doba od počátku zpracování konkrétního sloupce po jeho ukončení.

Tabulka 10.2: Zpracování statistik validačních dotazů

Sloupec	Typ	Čas	Různé	Prázdné
Dotaz D.1 - *CT vyšetření*				
stroke_id	xsd:string	5.83s	5 510	0 / 5 879
stroke_idSITS	xsd:string	0.8s	1 232	4 561 / 5 879
…ctDone…	xsd:boolean	0.58s	2	1 979 / 5 879
…ctDone…24h…	xsd:boolean	0.09s	2	3 983 / 5 879
…ctDateTime…	xsd:dateTime	4.38s	4 282	1 216 / 5 879
…ctDateTime…24h	xsd:dateTime	1.78s	2 099	3 510 / 5 879
prázdné sloupce	- - -	0.03s	- - -	5 879 / 5 879
Dotaz D.2 - *Akutní intervence*				
…anesthesiaType…	uri	0.14s	2	5 178 / 5 815
…occlusionLocalisation…	uri	0.03s	43	5194 / 5 815
Další sloupce číselníků a literálů...				
Dotaz D.3 - *Populace měst*				
city	uri	20.83s	9 936	0 / 10 000
country	uri	0.43s	234	0 / 10 000
population	xsd:integer	16.87s	9 021	0 / 10 000
Dotaz D.4 - *Přehled obrazových sérií*				
study_id	xsd:integer	57.52s	6 952	0 / 129 367
series_number	xsd:integer	21.69s	696	0 / 129 367
series_name	xsd:string	198.52s	38 019	0 / 129 367
study_name	xsd:string	20.75s	148	0 / 129 367

10.1.3 Zpracování metadat sloupce

Průměrné časy zpracování sloupce v tabulce 10.3 zahrnují operace automatického zpracování po zpracování statistik 10.1.2. Jedná se o přiřazení přednastavených vlast-

ností dle datového typu nebo predikátu a získání metadat pro IRI sloupce. Je uveden vzorek různých typů sloupců s rozdíly dle distribuce a datového typu.

Tabulka 10.3: Zpracování metadat validačních dotazů

Sloupec	Typ	Čas	Různé	Dotazy
Dotaz D.1 - *CT vyšetření*				
stroke_id	xsd:string	0.09s	5 510	- - -
stroke_idSITS	xsd:string	0.37s	1 232	- - -
…ctDone…	xsd:boolean	0.48s	2	- - -
…ctDone…24h…	xsd:boolean	1.38s	2	- - -
…ctDateTime…	xsd:dateTime	0.19s	4 282	- - -
…ctDateTime…24h	xsd:dateTime	0.46s	2 099	- - -
prázdné sloupce	- - -	0.05s	- - -	- - -
Dotaz D.2 - *Akutní intervence*				
…anesthesiaType…	uri	7.07s	2	1
…occlusionLocalisation…	uri	7.16s	43	1
Další sloupce číselníků a literálů...				
Dotaz D.3 - *Populace měst*				
city	uri	25.78s	9 936	10
country	uri	7.42s	234	1
population	xsd:integer	0.12s	9 021	- - -
Dotaz D.4 - *Přehled obrazových sérií*				
study_id	xsd:integer	0.12s	6 952	- - -
series_number	xsd:integer	1.19s	696	- - -
series_name	xsd:string	0.05s	38 019	- - -
study_name	xsd:string	0.63s	148	- - -

10.1.4 Výkonnost zpracování dat

Čas zpracování statistik sloupců značně roste s výskytem rozsáhlého množství odlišných hodnot a vyšším množství znaků v měřených hodnotách. Doba výpočtu statistik dosahovala extrémní hodnoty 198 sekund u sloupce *series_name* dotazu D.4, kdy sloupec obsahoval rozsáhlé množství různých hodnot (38 019), které byly zároveň dlouhými řetězci, což je stejné chování jako u IRI hodnot, které jsou pro výpočet statistik dlouhými řetězci. Sloupec *study_name* z dotazu D.4 pracuje také se 129 367 záznamy, ale doba měření díky množství 148 různých hodnot trvá téměř 10x kratší dobu 21 sekund.

Zpracování metadat sloupců literálů je téměř ve všech případech řešeno pod jednu sekundu. V případě, že se jedná o IRI sloupec, je doba dalšího zpracování závislá nejprve na počtu odlišných nalezených hodnot (počet dotazů na metadata) a posléze na době odezvy jednotlivých endpointů. Příkladem je zpracování *city* z dotazu D.3 s deseti dotazy a časem 25.78 sekund oproti country ze stejného dotazu, který potřebuje pouze jeden dotaz s časem 7.42s. Výkonnost transformací přidání sloupců z IRI hodnot se chová stejným způsobem jako zpracování jednoho IRI sloupce a stejnými parametry roste i jejich časová náročnost.

10.2 Validace implementovaných požadavků

Dle tabulky B.1, která obsahuje označení implementovaných požadavků na SW, budou jednotlivé body zvalidovány. Validace požadavků předpokládá stav, kdy je uživatel do aplikace přihlášen a přísluší mu výchozí SPARQl nastavení.

DZ1 - SPARQL: Uživatel přejde na obrazovku vytvoření zdroje dat. Jsou zadány údaje SPARQL dotazu D.3 a uživatel zvolí vytvořit zdroj. Probíhá animace načítání a po několika sekundách je zobrazena tabulka s daty.

DZ2 - Basic Auth: DZ1 s dotazem 6.1, adresou lokálního SPARQL endpointu a zvolením BASIC autorizace s příslušnými údaji. Probíhá animace načítání a po několika sekundách je zobrazena tabulka s odpovídajícími záznamy z autorizovaného lokálního úložiště Apache Jena Fuseki.

DZ3 - Digest Auth: DZ1 s dotazem D.1 a zvolením DIGEST autorizace s příslušnými údaji. Probíhá animace načítání a po několika sekundách je zobrazena tabulka s odpovídajícími daty.

TA1 - Pozice proměnných: DZ1 s dotazem D.3. Po zpracování se v aplikační databázi nachází uložené pozice sloupců: `population` - objekt predikátu *dbo:populationTotal*, `country` - objekt predikátu *dbo:country* a subjekt s predikátem *a*, `city` - subjekt s predikáty *a*, *dbo:country*, *dbo:populationTotal*.

TA3 - Zpracování statistik: DZ1 s dotazem D.3. Po zpracování se v aplikační databázi nachází zpracované statistiky jednotlivých sloupců. Počty odpovídají očekávaným hodnotám, které byly porovnány se zpracovanými statistikami v aplikaci Power BI Desktop (po konverzi dat do CSV).

TA4 - Mapování dat. typu: DZ1 s dotazem D.1. U sloupců *stroke_id*, *stroke_idSITS* je zobrazen rozpoznaný datový typ řetězce. Sloupec *stroke_ctDone...* přiřazuje booleovský datový typ a *stroke_ctDateTime...* datový typ datum a čas.

U sloupců s prázdnými hodnotami je otazníky indikováno, že nebyl rozpoznán žádný datový typ. Rozpoznané datové typy odpovídají datovým typům dat dle výchozích SPARQL nastavení.

TA5 - Mapování zobrazení literálu: DZ1 s dotazem D.1. Sloupec *stroke_ctDone...* s přiřazeným booleovským datovým typem obsahuje hodnoty ano a ne se zarovnáním doleva. Sloupec *stroke_ctDateTime...* s přiřazeným datovým typem datumu a času zobrazuje hodnoty ve formátu *21. března 2023 15h*. Zobrazení hodnot odpovídá nastavenému zobrazení datových typů.

TA7 - Získání vzorku IRI vlastností: DZ1 s dotazem D.2. Je otevřeno nastavení sloupce *stroke_thrombectomyInitialOcclusionTICIScore1...* s přiřazeným datovým typem IRI a zobrazeno menu *Sample IRI attributes*. V menu jsou vlastnosti: *form:vocabularyId*, *form:vocabularyLineNumber*, *rdfs:label*, *rdf:type*, *vs:term_status*, *form:vocabularyItemId* a *form:code* s hodnotami objektů. Vlasnosti odpovídají predikátům IRI subjektů ve sloupci zobrazených v aplikaci Virtuoso na dotazovaném úložišti.

TA8 - Získání nastavených IRI vlastností: DZ1 s dotazem D.2. Sloupec *stroke_thrombectomyInitialOcclusionTICIScore1...* s přiřazeným datovým typem IRI je zobrazen ve své textové podobě, která je získaná načtením výchozí vlastnosti *rdfs:label*.

TO1 - Přehled získaných dat: DZ1 s dotazem D.3. Tabulka zobrazuje prvních 200 záznamů získaných dat a po její pravé straně je posuvník, který umožňuje výsledky procházet. V pravém spodním rohu jsou tlačítka *LOAD MORE*, které do zobrazení přidávají dalších 500 řádků. *LOAD ALL* po prodlevě prohlížeče zobrazuje všechny získané hodnoty.

TO3 - Řazení: DZ1 s dotazem D.3. Je otevřeno nastavení sloupce *city* a zvoleno *Sort ascending*. Řádky jsou seřazeny vzestupně dle hodnoty IRI sloupce zobrazené v *rdfs:label*. U sloupce *city* je zobrazeno tlačítko řazení, které po stisknutí mění směr řazení. Je nastaveno vzestupné řazení dle sloupce *population*. Řazení u sloupce *city* mizí.

TO4 - Statistiky sloupce: DZ1 s dotazem D.3. Je otevřeno nastavení sloupce *population*. Spodní část nastavení zobrazuje tabulku se statistikami. Je možné přepínat statistiky typu, třídy, datového typu, hodnoty a jazyka. Statistiky lze řadit dle počtu nebo agregované hodnoty. Stisknutím tlačítka *LOAD MORE* je zobrazeno více záznamů. Statistiky odpovídají zpracovaným statistikám sloupců.

TO6 - Přehled použitých transformací: DZ1 s dotazem D.3. Je stisknuto tlačítko *TRANSFORMATIONS* a otevřen pohled transformací, kde je v pravé části obrazovky prázdná plocha aplikovaných transformací. Je otevřeno nastavení sloupce *country* a stisknuto tlačítko *Copy column*. Obrazovka se znovu načte a v pohledu transformací je zobrazen záznam *Copy column country*. Je stisknuto *Reset transformations*, obrazovka se znovu načítá a seznam transformací je prázdný.

DT1 - Celé číslo: DZ1 s dotazem D.3. Je otevřeno nastavení sloupce *population* a je mu nastaven datový typ *number*. Je otevřeno *Display settings* a nastavena hodnota nula čísel za desetinnou čárkou. Je stisknuto *Apply settings* a sloupec obsahuje pouze celočíselná zobrazení hodnot.

DT2 - Desetinné číslo: DT1 s nastavením hodnoty tří čísel za desetinnou čárkou. Sloupec obsahuje desetinné zobrazení hodnot.

DT6 - Datum a čas: DZ1 s dotazem D.1. Je otevřeno nastavení sloupce *stroke_ctDateTime...* a je mu nastaven datový typ *dateTime*. Je otevřeno *Display settings* a nastaven vstupní vzor datumu odpovídající *xsd:dateTime* - %Y-%M-%dT%h:%m:%s:%u+%t. Je stisknuto *Apply settings*, sloupec obsahuje hodnoty typu datum a čas a řazení hodnotám odpovídá.

DT7 - Datum: DT6 s nastavením výstupního vzoru %Y-%M-%d. Sloupec obsahuje hodnoty typu datum a řazení hodnotám odpovídá.

DT8 - Čas: DT6 s nastavením výstupního vzoru %h:%m:%s:%u+%t. Sloupec obsahuje hodnoty typu čas a řazení hodnotám odpovídá.

DT10 - Rok: DT6 s nastavením výstupního vzoru %Y. Sloupec obsahuje hodnoty typu rok a řazení hodnotám odpovídá.

DT12 - Text: DZ1 s dotazem D.1. Je otevřeno nastavení sloupce *stroke_id* a je mu nastaven datový typ *string*. Sloupec obsahuje hodnoty typu text a řazení hodnotám odpovídá.

DT13 - Boolean: DZ1 s dotazem D.1. Je otevřeno nastavení sloupce *stroke_ctDone...* a je mu nastaven datový typ *boolean*. Sloupec obsahuje hodnoty typu boolean a řazení hodnotám odpovídá.

DV1 - Zarovnání: DZ1 s dotazem D.1. Je otevřeno nastavení sloupce *stroke_ctDone...* a jeho nastavení vzhledu. Je nastaveno *Align content* na hodnotu *right* a stisknuto *Apply settings*. Zobrazení vykresluje hodnoty na pravé straně sloupce. Je nastaveno *center* a zobrazení vykresluje hodnoty uprostřed sloupce.

Je nastaveno *left* a vykreslení hodnot je nalevo. Je stisknuto *Save column settings*, uložení je nastaveno a po opětovném načtení zachováno.

DV2 - Název sloupce: DZ1 s dotazem D.1. Je otevřeno nastavení sloupce *stroke_ctDone...* a stisknuto *Set column name*. Je vyplněn nový název sloupce *CT datum* a stisknuto *Set name*. V záhlaví sloupce je zobrazen zvolený název.

DV3 - Šířka sloupce: DZ1 s dotazem D.1. Je otevřeno nastavení sloupce *stroke_ctDone...* a stisknuto *Display settings*. Je nastaveno *Column width* na 5 a stisknuto *Apply settings*. Šířka sloupce je změněna.

DV6 - Prefixové zobrazení IRI: DZ1 s dotazem D.2. Je otevřeno nastavení IRI sloupce *stroke_thrombectomyInitialOcclusionTICIScore1...* a rozbaleno nastavení zobrazení (*Display*), kde je zvoleno *iri*. V nastavení je stisknuto *View prefix form IRI* a sloupec je zobrazen v prefixové podobě. Je stisknuto *View full IRI* a sloupec je zobrazen v celé IRI podobě.

DV7 - Zobrazení vlastností IRI: DZ1 s dotazem D.2. Je otevřeno nastavení IRI sloupce *stroke_thrombectomyInitialOcclusionTICIScore1...* a rozbaleno nastavení zobrazení (*Display*), kde je zvoleno *rdfs:label*. Sloupec zobrazuje *rdfs:label* hodnot místo IRI.

DV8 - Zobrazení čísla: DT1 s nastavením zobrazení *Digits group per* 3. Je nastaveno zarovnání doprava, které se zobrazí. Číslice jsou seskupovány po třech. Je nastaveno zobrazení *Digits group separator* - a skupiny čísel jsou odděleny -. Je nastaveno zobrazení tří čísel za desetinnou čárkou a zobrazení tečky jako desetinného oddělovače. Čísla jsou zobrazena odpovídajícím způsobem.

DV10 - Zobrazení datumu: DT6 s nastavením výstupního vzoru %M a *Month display* english. Hodnoty ve sloupci jsou zobrazeny pouze hodnotami měsíce v anglickém jazyce. Je nastaveno zobrazení czech a vzor %d. %m. Hodnoty jsou zobrazeny v odpovídajícím formátu např. *2. července*.

DV11 - Zobrazení boolean: DT13 s nastavením zobrazení *True display* YES. Místo hodnot true je ve sloupci zobrazeno YES. Je nastaveno *False display* X. Místo false hodnot je ve sloupci zobrazeno X.

DF1 - RDF filtrace: DZ1 s dotazem D.3. Je otevřeno nastavení sloupce *country*. Spodní část nastavení zobrazuje tabulku se statistikami a filtracemi. Je možné přepínat filtry typu, třídy, datového typu, hodnoty a jazyka. Je zvoleno nastavení tříd a všechna označení jsou odstraněna. Je vybráno United States, India, Iran a stisknuto tlačítko *APPLY FILTERS*. V tabulce se nyní nachází 4 367 záznamů a všechny mají záznam země z odpovídající množiny. Je stisknuto tlačítko zrušení filtrace v záhlaví a jsou zobrazeny všechny hodnoty.

TO1 - Načtení IRI vlastností: TA5 a je stisknuto levé tlačítko bez ohraničení u vlastnosti *form:vocabularyId*. Obrazovka je znovu načtena a ve sloupci je možné zvolit zobrazení hodnoty *form:vocabularyId*. V seznamu transformací přibyl záznam *Add iri value ...*.

TO2 - Přidání sloupce z IRI vlastnosti: TA5 a je stisknuto pravé tlačítko s ohraničením u vlastnosti *form:vocabularyId*. Obrazovka je znovu načtena a na konci tabulky přibyl sloupec s odpovídajícími hodnotami. V seznamu transformací přibyl záznam *Add column from ...*.

TO3 - Odstranění sloupce: DZ1 s dotazem D.3. Je otevřeno nastavení sloupce *country* a stisknuto *Delete column*. Obrazovka je znovu načtena a sloupec se v tabulce nenachází. V seznamu transformací přibyl záznam *Delete column country*.

TO4 - Kopírování sloupce: TO6 zobrazuje před smazáním transformací zkopírovaný sloupec na konci tabulky.

VD1 - Více stránek: Uživatel přejde na obrazovku vytvoření dashboardu. Náchází se na výchozí stránce *New page*. Stránka je přejmenována na *domů*. Je přidána další stránka a uživatel je na ni přesměrován. Kliknutím na spodní panel a stránku *domů* je přesměrován zpět na původní stránku.

VD2 - Přetečení: VD1 a uživatel stiskne *Add graph* a zvolí *Bar chart*. Uživatel myší uchopí vytvořený vizuál za spodní pravý okraj a zvětší ho přes velikost stránky. U stránky se objeví posuvník, který umožňuje přetečení obsahu.

VD3 - Libovolná pozice: VD2 a vizuál je myší uchopen za tlačítko kotvy v pravém horním rohu. Vizuál je libovolně přesouván tažením myši po stránce dashboardu.

VD4 - Libovolná velikost: VD2 a je omezena minimální velikostí vizuálu.

VD6 - Filtrace stránky: VD2 a uživatel v pravém panelu stiskne *Add data source*. Zvolí zdroj vytvořený DZ1. Uživatel stiskne nastavení vizuálu v jeho pravém horním rohu. V pravém panelu je k ose X přetažen sloupec *country* a k ose Y sloupec population, u nějž je zvolena agregace *Sum*. V pravém panelu s nastavením zdrojů je otevřeno menu sloupce *country* a je použita filtrace DF1. Graf nyní obsahuje pouze populace měst z filtrovaných států.

VD9 - Základní grafy: VD6 a uživatel v nastavení vizuálu zvolí typ grafu *Bar Chart*. Je vykreslen sloupcový graf populace filtrovaných zemí. Je zvolen typ grafu *Pie Chart* a obdobně je vykreslen koláčový graf. Stejně vizualizace pracuje i pro spojnicový graf, koblihový graf a polární diagram.

VI1 - Interaktivní nastavení hodnot: VD6 kliknutím na tlačítko koš v nastavení os je pole smazáno. Uživatel přidá další zdroj dat z dotazu D.2. Jeho sloupec je přetažen do nastavení vizuálu a sloupce v jiných polích, které přísluší jinému zdroji, jsou odstraněny.

VI2 - Agregace osy Y: VD6 a uživatel nastaví pro populaci agregaci *Sum*. Uživatel přetáhne sloupec *population* znovu do Y axis a tentokrát nastaví agregaci *Max value*. Ve vizuálu jsou zobrazeny pro každou kategorii nyní hodnoty *max - population* i *sum - population*.

VI3 - Kategorie osy X: VD6 a uživatel přetáhne myší sloupec *city* do nastavení osy X. Nyní jsou hodnoty shlukovány dle kombinace jména země a státu.

VI4 - Název grafu: VD6 a uživatel stiskne tlačítko úpravy u textu *Set title*. Nastaví název grafu *Populace* a stiskne tlačítko ukončení úpravy. Na horní části vizuálu je zobrazen tučný nápis s názvem grafu.

VI5 - Popisky os: VI4 a uživatel stiskne tlačítko úpravy u popisu osy Y *Set Y axis label*. Nastaví název osy Y *počet obyvatel* a stiskne tlačítko ukončení úpravy. Podél osy Y je zobrazen nápis *počet obyvatel*. Stejným způsobem je nastaven popis osy X a pod ní je zobrazen nápis *Země*.

VI7 - Detail IRI hodnoty: DZ1 s dotazem D.2. Uživatel klikne na libovlnou hodnotu sloupce *...thrombectomyInitialOcclusionTICIcore1...* a otevře se nová obrazovka. Obrazovka obsahuje všechny vlastnosti daného IRI subjektu a jsou čarou rozděleny na přednastavené uživatelem a ostatní. Kliknutím na jinou IRI hodnotu v přehledu je okno přesměrováno na detail dalšího IRI.

11 Diskuze

Automatické zpracování SPARQL dat přináší relevantní výsledky a značné usnadnění práce se SPARQL daty. Jsou využívány specifické možnosti propojených dat. Návrh aplikace vychází z vlastností analyzovaných aplikací. Implementace metody zpracování dat prokazuje použitelnost navrženého konceptu a nastiňuje i další rozsáhlý směr, kterým se vývoj aplikace může ubírat. Komplexní aplikace prakticky implementuje obrazovku pro aktivní transformaci propojených dat a jejich průzkum v tabulkové podobě, dashboardy i interaktivní vizualizace.

Aplikace v rozsahu přizpůsobení vzhledu dashboardů a množství použitelných vizuálů, grafických prvků nebo interaktivních ovládacích prvků nemůže analyzovaným aplikacím konkurovat. To samé platí pro množství specifických transformací aplikovatelných na jednotlivé datové typy, nebo dokonce celé řádky dat. Oproti ostatním aplikacím je snaha využít možnosti propojených dat ze SPARQL Endpointu a z nich vycházejících možností, které ostatní aplikace pracující s běžnými daty nemají. Automatické zpracování datových typů sloupců a jejich zobrazení, nebo dokonce získání alternativních vlastností IRI sloupců a práce s nimi, je vlastností, kterou aplikace svou konkurenci v případě práce s těmito daty značně převyšuje. Implementovaná aplikace má zejména na klientské části možnosti optimalizace a vylepšení.

11.1 Porovnání s analyzovanými aplikacemi

Hlavní konkurenční výhoda oproti ostatním nástrojům a nedostatečná podpora SPARQL Endpointu u ostatních aplikací, je podnětem realizace diplomové práce. Nejsilnější podpora SPARQL Endpointu mezi analyzovanými nástroji byla v aplikaci Looker Studio, a to pouze v podobě komunitního konektoru. Pro ostatní aplikace byla práce s RDF daty v jakékoliv podobě omezená. Jediná možnost získání dat v implementované aplikaci je právě SPARQL Endpoint. Z nastaveného připojení k endpointu a úložišti ontologií jsou i zpracovávána metadata.

Návrh obrazovky transformací dat vycházel nejvíce z transformační obrazovky aplikace Power BI, oproti které jsou zde plně podpořeny rozdílné podoby zobra-

zení IRI hodnot a práce s nimi, nebo úprava zobrazení literálů. Aplikace rozlišuje jen základní datové typy řetězec, datum a čas, číslo, boolean a navíc IRI hodnoty. Chybí zde například podpora geografických údajů nebo URL hodnot odkazující na obrázky, jejichž podpora se v analyzovaných aplikacích vyskytovala. V rámci specifikace požadavků tyto datové typy neměly kritickou prioritu a jejich podporu lze doimplementovat.

Narozdíl od ostatních analyzovaných nástrojů jsou v aplikaci i uživatelská nastavení, která ovlivňují, jakým způsobem budou získávaná data zpracovávána a zobrazována. Jedná se o předvolby zobrazení datových typů, získaných vlastností IRI, nebo jazykové preference a prefixy. Výhoda je umožněná díky tomu, že se jedná o propojená data.

Transformační obrazovka datového zdroje je koncipována jako průzkumná, filtry fungují stylem, že s nimi uživatel provádí interakci a v případě vůle filtraci aplikuje a filtrovaná data jsou odstraněna. Filtrační pravidla dle datového typu jako větší, menší nebo rovno apod. implementovány nebyly, jsou na seznamu požadavků. Úprava filtrací pracuje s menu podobným aplikaci Power BI, kde uživatel vidí rozložení daných hodnot v rámci sloupce, může rychle vyhodnotit povahu daného sloupce a nevyžádané hodnoty zahodit. To platí i pro filrovatelné souhrny dle typu, datového typu, hodnoty, IRI hodnoty nebo jazyka. Transformované hodnoty lze navíc exportovat pro libovolné další použití v jiných nástrojích.

Aplikace implementuje základní transformace přidání sloupce, odebrání sloupce, kopírování sloupce a filtrace hodnot. Dále se již jedná o IRI transformace, přidání IRI vlastností jako sloupce nebo jako použitelnou vlastnost sloupce. Široká nabídka transformací pro specifické datové typy jako u Power BI zatím implementována není.

Uchování transformačních kroků a jejich aplikace vychází z inspirace u Power BI. Všechny transformace jsou aplikovány po znovunačtení dat a lze je odstraňovat. Oproti Power BI zatím není dovoleno mezi transformacemi přecházet, měnit jejich zobrazení, pořadí, sledovat statistiky ovlivněných řádků nebo čas vykonání apod.

Aplikace využívá rozložení dashboardů jako například Looker Studio, kdy je dashboard rozdělen na jednotlivé stránky a na ně libovolně umisťuje libovolný počet vizuálů. Každá stránka může používat libovolné zdroje dat, kdy jeden vizuál má data pouze z jednoho zdroje dat, jak je běžné i u ostatních aplikací. V aplikaci jsou k dispozici pouze základní vizuály sloupcový graf, spojnicový graf, koláčový graf, koblihový graf a polární diagram. Ovládací prvky, grafické prvky nebo možnosti barevného přizpůsobení vizuálů a stránek nejsou oproti ostatním nástrojům podporovány.

Obdobně jako u ostatních aplikací mají jednotlivé vizuály definované své atributy, kterým se přiřadí konkrétní sloupce ze zdroje dat. Sloupec se z pravého panelu zdroje dat přetáhne myší na příslušné pole. Osa X slouží jako seskupovací a osa Y

pro výpočet, což odpovídá povaze nastavení worksheetů aplikace Tableau Desktop, kde se hodnoty osy X nazývají dimenze a hodnoty osy Y míry. Jednotlivé hodnoty osy Y nabízejí možnosti agregací, které jsou obdobné agregacím u vizuálů v Power BI. Dashboard umožňuje filtrace dat pro celou stránku, kdy jsou ovlivněny všechny vizuály na dané stránce. Podpora vlastní filtrace vizuálů, která je umožněna u ostatních nástrojů, implementována není, ale podmínky pro její implementaci jsou v aplikaci připraveny.

Kliknutí na konkrétní záznam ve vizuálu v aktuální implementaci ostatní vizuály neovlivňuje. Je možné pouze filtrovat data pro danou stránku, kdy jsou všechny vizuály okamžitě ovlivněny. Při tažení myši přes vizuál je zobrazena konkrétní hodnota zobrazeného záznamu. V aplikaci je implementován vyskakovací pohled na IRI hodnoty, který otevře nové okno, kde lze všechny vlastnosti zvolené IRI hodnoty prozkoumávat. To je jedna z možností SPARQL dat, která opět využívá výhody propojených dat a sdíleného schéma v podobě slovníku a ontologií.

Sdílení dashboardů implementováno nebylo, což je nevýhodou oproti ostatním nástrojům. Nicméně bylo na problematiku myšleno a je připraven klientský pohled bez menu, který se využívá například u vyskakovacího zobrazení detailu IRI.

11.2 Omezení aplikace

Validační část výkonnosti automatického zpracování dat (viz 10.1) ukazuje, že může v aplikaci nastat stav, kdy bude výpočetní doba zpracování statistik sloupce výrazně stoupat. Vhodná optimalizace použitých datových struktur (momentálně je použit základní pythonový *dictionary*), vhodné hashování dlouhých názvů tříd nebo dlouhých řetězců, může problém vyřešit. Extrémní doba zpracování trvající 198.52 sekund nastala pouze u sloupce *series_name* dotazu D.4 a ve sloupci se nacházelo 38 019 odlišných hodnot. Obdobný problém zvětšování náročnosti výpočtu metadat byl změřen u zpracovávání IRI sloupců a vykonávání jejich SPARQL dotazů. Záleží na odezvě dotazovaných endpointů. Vhodná implementace cachování získávaných metadat by mohla omezit dobu odezvy v případě častého používání množství podobných dat. Případně může pomoci vhodný import relevantních slovníků a ontologií do RDF úložiště, ze kterého se očekává rychlejší odezva než z externích endpointů, které můžou reagovat na dotazy s různou rychlostí.

Možným problémem zpracovávání IRI sloupců může být nedostupnost požadovaných metadat. Pokud nejsou vlastnosti daných IRI subjektů dostupné ani na dotazovaném SPARQL Endpointu, ani ve vlastním úložišti, nezbývá, než aby se uživatel spokojil se zkrácenou prefixovou podobou daných URI. Vzhledem k povaze propojených dat by samozřejmě mělo být možné se k slovníkům a ontologiím dat dostat a nahrát je do vlastního úložiště, ale ne vždy to musí být přímočaré a ne vždy budou dané slovníky dostupné.

Jak bylo ukázáno například u dotazu D.3 z endpointu *dbpedia*, může nastat stav, kdy endpoint vrací na dotaz nezávisle na reálnému počtu odpovídajících dat pouze omezené množství dat. U zkoumaného dotazu konkrétně maximálně 10 000 záznamů. Jedná se o problém, který by aplikace mohla řešit v případě, že daný stav rozpozná, nebo je uživatelem oznámen. Pro danou stiuaci je vhodná konstrukce LIMIT a OFFSET s následným dávkováním daného dotazu na endpoint.

11.3 Možnosti rozšíření aplikace

11.3.1 Uživatelské rozhraní aplikace

Prvním bodem rozšíření je optimalizace interakce s uživatelským rozhraním klienta aplikace. Testování prokázalo řadu nedostatků interakce, například s uživatelskými menu. Zásadní problémy byly ihned opraveny, ale na řadu detailů nezbyl prostor. Klientská část aplikace je komplexní a v rámci vývoje byl kladen důraz na splnění funkčních požadavků na úkor detailů, které by bylo vhodné v případě dalšího rozšiřování aplikace vyřešit.

Možnosti rozšíření nastávají u obrazovky dashboardů, kde přichází v úvahu přidání množství dalších grafů, vizuálních prvků nebo i interaktivních ovládacích prvků. Rozmisťování těchto prvků může být rozšířeno o úchytné osy a asistenci zarovnání celé stránky dashboardu. Dále je možné rozšíření možností přizpůsobení vzhledu a nastavení jednotlivých vizuálů. Vhodná je implementace křížového filtrování vizuálů na jedné stránce, kde by kliknutí na hodnotu mohlo ovlivňovat filtry zvolené stránky, které jsou již implementované. Mohou být doplněny i filtry jednotlivých vizuálů nebo nastavení, zda je vizuál ovlivněn dalšími filtry na stránce a jak má ostatní filtry ovlivňovat.

Pro menší závislost na znalostech RDF a SPARQL by bylo vhodné místo všech IRI hodnot vlastností, se kterými uživatel pracuje, zobrazovat jejich popisek. Laik tak snadno uchopí práci s IRI sloupci, které mohou být pojaty jednoduše jako objekty, které mají své vlastnosti, s nimiž lze pracovat. Funkcionalita získání IRI metadat na serverové části již získávání *rdfs:label* všech dotazovaných vlastností podporuje a je využíváno například v zobrazení detailu IRI subjektu, kde jsou vypsány jeho vlastnosti i s textovým označením vlastností. Umožnění takového chování vyžaduje přístup k relevantním ontologiím a slovníkům, které je nutné importovat.

11.3.2 Metoda zpracování dat

Možností vylepšení zpracování dat je implementace podpory pro soubory všech RDF formátů jako zdroje dat. Přichází v úvahu i obyčejné CSV soubory, které obsahují URI, jež budou aplikací rozpoznány nebo uživatelem nastaveny jako URI

sloupce a do dané podoby převedeny. Zpravidla stačí, aby aplikace dokázala konvertovat vstup do podoby v JSON-LD. Ve scénáři, kdy je vstupem soubor, je vhodné poskytnout možnost zadání SPARQL Endpointu pro zpřístupnění získání metadat IRI sloupců.

Možností vylepšení samotné metody automatického zpracování dat a jejich sloupců je celá řada, například automatické získání prefixů z uživatelských dotazů, které mohou být importovány do seznamu uživatelových prefixů. Zpracování prefixů ze SPARQL je již na serverové části implementováno.

V serverové části aplikace je pro získání vlastností sloupce implementováno rozpoznání pozice proměnných z dotazu. Může poskytnout *rdfs:domain* nebo *rdfs:range*, které určí název a typ sloupce.

Obrazovka pro nastavení vlastností IRI sloupců dle rozpoznaného predikátu je implementována i na klientské části aplikace, ale z důvodu nižší priority nebylo řešení dokončeno. Pokud by se přímo v rámci obrazovky daly získávat vlastnosti predikátů pro konkrétní endpoint, mohly by sloužit pro přednačtení dat sloupce, kdy by při samotném zpracování IRI sloupce stačilo doplnění chybějících hodnot, které nebyly předem načteny.

Získání vlastností IRI sloupce na serverové části aplikace běží, jen není dostupná podpora na straně klienta, která by dané nastavení umožňovala vytvářet. V případě například manuálního přidání položek do databáze bude aplikace nastavení využívat. Implementace uživatelského nastavení IRI sloupců dle predikátů by mohla rozšířit sílu automatické metody zpracování, ať už o přednastavení načítaných vlastností, nebo o získání odpovídajícího názvu a popisku sloupce dle rozpoznané vlastnosti z predikátu.

Poslední zmíněnou částí vylepšení automatické metody zpracování dat jsou širší možnosti nastavení vlastních RDF úložišť ontologiií a slovníků. Nabízí se implementace automatického importu úložišť v případě, kdy aplikace vyhodnotí, že daný slovník nebo ontologie obsahuje požadovaná metadata a není v úložišti přítomen. Správa nastavení importů nebo úložiště by mohla být implementována v uživatelském rozhraní aplikace.

11.3.3 Integrace s dalšími nástroji

Integrace s dalšími nástroji může být implementace exportu transformovaných dat z JSON-LD do dalších formátů (CSV, běžný JSON apod.), se kterými umí pracovat i ostatní aplikace. V rámci uživatelského rozhraní je implementována možnost stažení zdroje dat, které však poskytuje data zatím pouze v původním formátu JSON-LD. Aplikace může ke svým transformovaným datům poskytovat i API volání, jež je rovněž na serverové části implementováno. Pomocí API by mohly ostatní aplikace transformované zdroje SPARQL dat periodicky získávat. Tímto způsobem

by byla využita hlavní silná stránka implementované aplikace (zpracování a transformace SPARQL a RDF dat) a v dalších aplikacích by byla data využívána pro další transformace, vizualizace, nebo jiné činnosti související s datovou analýzou.

Lze integrovat s nástroji, které produkují SPARQL dotazy. Příkladem může být plánovaná integrace se SPARQL Query-Builder [41], která umožňuje interaktivní tvorbu dotazů. Obdobně lze integrovat aplikaci s projektem Sparkle, který pracuje s metadaty endpointu již při tvorbě dotazu [42]. Oba SW pocházejí z KIV. Vytvoření a úprava SPARQL dotazů je v aktuální implementaci aplikace pouze obyčejným textovým polem a přichází v úvahu, aby ji komponenta dobře integrovaného nástroje pro tvorbu SPARQL dotazů zcela nahradila.

11.3.4 Statistická analýza

Metody statistické analýzy nad transformovanými zdroji dat mohou být podpořeny implementací komponent pro zkoumání charakteristik, vzorů, trendů dat nebo aplikování statistických testů. Jelikož je serverová část aplikace implementována v jazyce Python, jsou možnosti implementace těchto metod statistické analýzy rozsáhlé, a existuje i možnost integrace s moduly jazyka R.

Závěr 12

V práci jsou popsána propojená data a sémantické technologie RDF, ontologie, slovníky, jazyk SPARQL a možnosti SPARQL endpointu. Jsou zkoumány procesy analýzy a vizualizace dat, porovnány aplikace Power BI, Looker Studio, Tableau a JavaScriptové vizualizační knihovny. Na základě analytické části je navržena metoda pro podporu zpracování propojených dat, která je rozdělena na získání dat, automatickou transformaci, manuální transformaci, vizualizaci, interakci s vizuály a dashboardy. Je definována specifikace požadavků na SW, návrh aplikační databáze a jsou zvoleny vhodné technologie pro implementaci.

Vzniká klient-server aplikace, která umí propojená data v RDF ze SPARQL Endpointu interaktivně vizualizovat v srozumitelné a přehledné podobě. K tomu slouží automatické i manuální (interaktivní) transformace, které surová RDF data zpracují do přehledných tabulek se sloupci formátovanými dle datového typu nebo dalších požadavků uživatele. Data lze dále prezentovat formou vícestránkových dashboardů, které obsahují několik typů vizuálů. Každý zpracovaný sloupec je doplněn o statistický přehled složení hodnot a jejich typů a lze intuitivně filtrovat. K zpracování sloupců využívá aplikace RDF slovníky a ontologie, ze kterých získává metadata v souladu s jazykovými preferencemi uživatele. *Uri* sloupce tak mohou být zobrazeny v lidsky čitelné podobě pomocí *rdfs:label*, nebo libovolné vlastnosti, která je pro ně, manuální transformací nebo automaticky dle přednastavení uživatele, získána.

Aplikace je otestována jednotkovými testy proti lokálnímu SPARQL endpointu a všechny obrazovky jsou důkladně manuálně otestovány na straně klienta. Aplikace je validována na netriviálních dotazech, je změřena výkonnost zpracování dat, validováno splnění specifikovaných požadavků a nakonec je vzniklá aplikace porovnána s analyzovanými aplikacemi. Práce je doplněna o uživatelskou příručku, popis databázového modelu a v elektronické příloze dokumentaci serverového REST API.

Elektronické přílohy

A

Obsah souboru `A22N0050P_prilohy.zip`, jenž je přiložen k diplomové práci, obsahuje několik adresářů a souborů:

- Obsah adresáře `Aplikace a knihovny` je složen ze dvou podadresářů - `api` a `app`.
  - Adresář `api` obsahuje dokumentaci REST API volání serverové části aplikace. Dále jsou zde obsaženy soubory pro import těchto API volání do aplikace Postman a Swagger.
  - Adresář `api` obsahuje zdrojový kód aplikace a manuál jejího sestavení. Kód serveru je pak v podadresáři `backend_flask` a kód klienta v podadresáři `frontend_react`.
- Adresář `Poster` obsahuje všechny soubory k vytvořenému posteru.
- Obsahem adresáře `Text_prace` jsou pak všechny zdrojové soubory pro překlad diplomové práce do PDF a výsledek překladu s textem diplomové práce.
- Adresář `Vstupni_data` je složen z následujících podadresářů:
  - Adresář `test` obsahuje testovací data pro jednotkové testy.
  - Adresář `validacni_dotazy` obsahuje SPARQL dotazy pro validaci.

Specifikace požadavků

B

Tabulka B.1 zobrazuje přehled požadavků na SW vzniklých z navržené metody. Požadavky jsou rozděleny dle priority (**P**) na *kritické - A*, *možná rozšíření - B* a *vhodná budoucí rozšíření - C*. Jsou shlukovány dle rozdělení procesů z části 4, čemuž přísluší i kódové označení (**C**). V rámci omezení opakování informace je v tabulce uvedeno i označení implementace daných požadavků (**I**), kde ✓ značí plně implementovaný stav, ✗ nezpracovaný požadavek a - částečně implementovanou funkcionalitu.

Tabulka B.1: Specifikace požadavků na SW

C	Účel	Označení	P	I
DZ - *Získání dat*				
DZ1	Získání dat ze SPARQL Endpointu	SPARQL	**A**	✓
DZ2	DZ1 s autentizací Basic Auth	Basic Auth	**B**	✓
DZ3	DZ1 s autentizací Digest Auth	Digest Auth	**B**	✓
DZ4	DZ1 s tokenovou autentizací	Token Auth	**C**	✗
DZ5	DZ1 s autentizací OAuth	OAuth	**C**	✗
DZ6	Získání dat ze souboru v RDF formátu	Soubor RDF	**B**	✗
DZ7	Periodická aktualizace dat z endpointu	Aktualizace dat	**B**	✗
TA - *Automatická transformace*				
TA1	Zpracování pozic ze SPARQL dotazu	Pozice proměnných	**B**	✓
TA2	Získání metadat z pozice proměnné	Metadata dle pozice	**B**	✗
TA3	Zpracování statistik hodnot ve sloupci	Zpracování statistik	**A**	✓
TA4	Přiřazení datového typu sloupci literálů dle statistik	Mapování dat. typu	**A**	✓
TA5	Přiřazení vykreslení sloupce dle dat. typu	Mapování zobrazení literálu	**B**	✓

(tabulka pokračuje na další stránce)

Tabulka B.1 (*pokračování z předchozí stránky)*

C	Požadavek	Označení	P	I
TA6	Přiřazení vykreslení IRI sloupce dle rozpoznaného predikátu	Mapování zobrazení IRI	**B**	-
TA7	Získání souhrnu vlastností IRI hodnot ve sloupci	Získání vzorku IRI vlastností	**A**	✓
TA8	Získání nastavených IRI vlastností pro hodnoty ve sloupci	Získání nastavených IRI vlastností	**A**	✓
Manuální transformace				
TO - Transformační obrazovka				
TO1	Průzkum získaných dat po jejich zpracování	Přehled získaných dat	**A**	✓
TO2	Průzkum libovolné části získaných dat	Stránkování	**B**	-
TO3	Řazení přehledu získaných dat dle sloupců	Řazení	**B**	✓
TO4	Zobrazení statistik sloupce	Statistiky sloupce	**A**	✓
TO5	Zobrazení statistik sloupce v grafické podobě	Vizualizace statistik sloupce	**B**	✗
TO6	Zobrazení a úprava použitých transformačních kroků	Přehled použitých transformací	**B**	✓
DT - Datové typy				
DT1	Podpora záznamů celých čísel	Celé číslo	**A**	✓
DT2	Podpora záznamů desetinných čísel	Desetinné číslo	**A**	✓
DT3	Podpora záznamů procenta	Procento	**B**	✗
DT4	Podpora záznamů doby trvání	Doba trvání	**B**	✗
DT5	Podpora záznamů měny	Měna	**B**	✗
DT6	Podpora záznamů datumu a času	Datum a čas	**A**	✓
DT7	Podpora záznamů s datumem	Datum	**A**	✓
DT8	Podpora záznamů času	Čas	**A**	✓
DT9	Podpora záznamů časového pásma	Časové pásmo	**B**	✗
DT10	Podpora záznamů roku	Rok	**A**	✓
DT11	Podpora záznamů čtvrtletí	Čtvrtletí	**B**	✗
DT12	Podpora textových záznamů	Text	**A**	✓
DT13	Podpora boolean záznamů	Boolean	**A**	✓
DT14	Podpora adresy URL	Adresa URL	**C**	✗
DT15	Podpora URL obrázku	URL obrázku	**C**	✗

(tabulka pokračuje na další stránce)

Tabulka B.1 (*pokračování z předchozí stránky)*

C	Požadavek	Označení	P	I
DT16	Podpora geografických záznamů	Geografické údaje	**C**	✗
DV - Přizpůsobení vzhledu sloupce				
DV1	Zarovnání obsahu sloupce	Zarovnání	**B**	✓
DV2	Změna názvu sloupce	Název sloupce	**A**	✓
DV3	Úprava šířky sloupce	Šířka sloupce	**B**	✓
DV4	Změna grafického zobrazení textu záznamů	Přizpůsobení fontu	**C**	✗
DV5	Zvýraznění záznamů dle pravidel	Zvýraznění záznamů	**C**	✗
DV6	Zobrazení IRI hodnot v prefixové podobě	Prefixové zobrazení IRI	**A**	✓
DV7	Zobrazení vlastnosti IRI hodnot místo IRI podoby	Zobrazení vlastnosti IRI	**A**	✓
DV8	Přizpůsobení zobrazení číselných hodnot	Zobrazení čísla	**B**	✓
DV9	Přizpůsobení zobrazení řetězce	Zobrazení řetězce	**B**	✗
DV10	Přizpůsobení zobrazení datumu a času	Zobrazení datumu	**B**	✓
DV11	Přizpůsobení zobrazení boolean hodnot	Zobrazení boolean	**B**	✓
DF - Filtrace				
DF1	RDF filtrace dle hodnoty, typu, datového typu, tříd a jazyka	RDF filtrace	**A**	✓
DF2	Filtrační pravidla čísel	Číselné filtrace	**B**	✗
DF3	Filtrační pravidla datumu a času	Filtrace datumu	**B**	✗
DF4	Filtrační pravidla pro řetězce	Filtrace řetězce	**B**	✗
TO - Operace				
TO1	Načtení vlastností pro IRI hodnoty sloupce	Načtení IRI vlastnosti	**A**	✓
TO2	Přidání sloupce z hodnot IRI vlastnosti	Přidání sloupce z IRI vlastnosti	**A**	✓
TO3	Odstranění sloupce	Smazání sloupce	**A**	✓
TO4	Zkopírování sloupce do nového	Kopírování sloupce	**B**	✓
TO5	Agregace dle kritéria	Seskupení hodnot	**B**	✗
TO6	Výpočet nových hodnot dle vzorce	Výpočet pole	**C**	✗

(tabulka pokračuje na další stránce)

Tabulka B.1 (*pokračování z předchozí stránky*)

C	Požadavek	Označení	P	I
TO7	Rozdělení sloupce do více sloupců dle kritéria	Rozdělení sloupce	**C**	✗
TO8	Nahrazení hodnot ve sloupci dle kritéria	Nahrazení hodnot	**C**	✗
TM - Modelování dat				
TM1	Spojení více zdrojů dat dle kritéria	Spojování zdrojů dat	**C**	✗
TM2	Podpora polí nebo hierarchických objektů	Hierarchie dat	**C**	✗
VD - *Dashboard*				
VD1	Rozložení dashboardu s více stránkami	Více stránek	**A**	✓
VD2	Rozložení dashboardu umožňující přetečení obsahu	Přetečení	**B**	✓
VD3	Libovolné přizpůsobení pozice vizuálu tažením myší	Libovolná pozice	**A**	✓
VD4	Libovolné přizpůsobení velikosti vizuálu tažením myší	Libovolná velikost	**A**	✓
VD5	Grafická podpora zarovnání umístění vizuálů v dashboardu	Asistence zarovnání	**C**	✗
VD6	Filtrace zdroje dat pro celou stránku dashboardu	Filtrace stránky	**A**	✓
VD7	Filtrace zdroje dat jen pro vizuál	Filtrace vizuálu	**B**	✗
VD8	Křížová filtrace zdroje dat kliknutím na prvek ve vizuálu	Křížová filtrace	**B**	✗
VD9	Podpora základních typů grafů	Základní grafy	**A**	✓
VD10	Podpora ovladacích prvků pro interakci s dashboardem	Ovládací prvky	**C**	✗
VD11	Podpora grafických prvků pro přizpůsobení vzhledu dashboardu	Grafické prvky	**C**	✗
VD12	Barevné přizpůsobení vzhledu dashboardu	Přizpůsobení dashboardu	**C**	✗
VI - *Interakce s vizuály*				
VI1	Přiřazení atributů vizuálu tažením sloupců myší	Interaktivní nastavení hodnot	**B**	✓

(*tabulka pokračuje na další stránce*)

Tabulka B.1 (*pokračování z předchozí stránky)*

C	Požadavek	Označení	P	I
VI2	Výpočet agregací dle hodnot v ose Y	Agregace osy Y	**A**	✓
VI3	Shlukování hodnot dle kategorií na ose X	Kategorie osy X	**A**	✓
VI4	Přizpůsobení názvu grafu	Název grafu	**B**	✓
VI5	Přizpůsobení popisků os	Popisky os	**B**	✓
VI6	Přizpůsobení barevného zobrazení grafu	Přizpůsobení grafu	**B**	✗
VI7	Zobrazení detailního pohledu po kliknutí na IRI hodnotu	Detail IRI hodnoty	**B**	✓
VS - *Sdílení dashboardu*				
VS1	Publikace dashboardu a jeho sdílení pomocí veřejného URL	Sdílení dashboardu	**B**	✗

Databázový model C

Základní rozdělení entit databáze z modelu na obrázku 7.1 lze rozlišovat následujícím způsobem: uživatel, SPARQL nastavení, zdroje dat a dashboardy. Jednotlivé skupiny databázových entit, včetně jejich účelu a atributů, budou následně blíže popsány. Pokud se jedná o povinný atribut, bude u jeho datového typu doplněn znak *.

C.1 Uživatel

Pro uživatele existuje aktuálně jedna základní entita *User*. Entita je v samotném středu databázového schématu. Představuje totiž jednoho uživatele aplikace, který je vlastníkem svých zdrojů dat, dashboardů a SPARQL nastavení. Uživatel má následující atributy:

Tabulka C.1: Popis entity User

Atribut	Typ	Popis
id	ID*	ID uživatele
login	String*	Přihlašovací jméno
password	String*	Zašifrovaná podoba hesla
name	String	Zobrazované jméno
lastActivity	DateTime	Poslední zaznamená aktivita
created	DateTime	Datum a čas registrace
prefLangs	String[]	Seznam preferovaných jazyků
defaultIriPredicates	String[]	Seznam výchozích IRI predikátů
authToken	String*	Aktuální autentizační token

C.2 SPARQL nastavení

V rámci databáze jsou navrženy pro podporu ukládání uživatelských SPARQL nastavení entity: *SparqlPrefix*, *SparqlDatatypeProperty*, *SparqlIriProperty*. V následující části budou vlastnosti jednotlivých entit blíže popsány.

C.2.1 SparqlPrefix

Entita *SparqlPrefix* představuje jeden uložený prefix uživatele a jeho odpovídající *uri* hodnotu.

Tabulka C.2: Popis entity SparqlPrefix

Atribut	Typ	Popis
userId	ID*	ID vlastníka prefixu
uri	String*	Úplná URI podoba
prefix	String*	Prefixová podoba

C.2.2 SparqlDatatypeProperty

Entita *SparqlDatatypeProperty* představuje jedno uložené uživatelské nastavení datových typů dle převažující třídy datového typu.

Tabulka C.3: Popis entity SparqlDatatypeProperty

Atribut	Typ	Popis
userId	ID*	ID vlastníka nastavení
uri	String*	URI datového typu
datatype	DataType* (C.5.5)	Přiřazený aplikační datový typ
exampleValue	EntryJsonLD (C.6.8)	Ukázková hodnota
displaySettings	ColumnDisplay (C.6.10)	Nastavení zobrazení

C.2.3 SparqlIriProperty

Entita *SparqlIriProperty* představuje jedno uložené uživatelské nastavení datových typů dle rozpoznané hodnoty IRI predikátu. Nastavení zároveň obsahuje předzpracovaná data vzorku hodnot a získaných vlastností.

Tabulka C.4: Popis entity SparqlIriProperty

Atribut	Typ	Popis
userId	ID*	ID vlastníka nastavení
uri	String*	URI predikátu
properties	ColumnProperties (C.6.9)	Nastavení sloupce
samplePredicates	ColumnSamplePredicates (C.6.15)	Získaný vzorek vlastností
iriValues	ColumnIriValues (C.6.14)	Získané vlastnosti IRI hodnot

C.3 Zdroj dat

V rámci databáze jsou s jedním zdrojem dat (*DataSource*) provázány tři další entity: *Data, DataColumn, DataTransformation*. V následující části budou blíže popsány.

C.3.1 DataSource

Entita představuje jeden zdroj dat v rámci aplikace a jeho nastavení. Může se jednat o zdroj SPARQL dotazu a jeho nastavení nebo zdroj popisující soubor, ze kterého je vytvořen.

Tabulka C.5: Popis entity DataSource

Atribut	Typ	Popis
id	ID*	ID zdroje dat
userId	ID*	ID vlastníka zdroje dat
name	String*	Zobrazované jméno zdroje

(tabulka pokračuje na další stránce)

Tabulka C.5 *(pokračování z předchozí stránky)*

Atribut	Datový typ	Popis
type	SourceType* (C.5.3)	Typ zdroje dat
sourceObject	SourceObject* (C.6.6)	Nastavení zdroje dat
lastUsed	DateTime	Datum a čas poslední zaznamenané aktivity
lastUpdated	DateTime	Datum a čas poslední zaznamenané změny
created	DateTime	Datum a čas registrace uživatele

C.3.2 Data

Entita *Data* představuje jeden soubor získaných a zpracovaných dat příslušící k jednomu konkrétnímu zdroji dat.

Tabulka C.6: Popis entity Data

Atribut	Typ	Popis
dataSourceId	ID*	ID zdroje dat
rows	Integer	Počet řádek
fetchDuration	Float	Doba získávání dat
lastUsed	DateTime	Datum a čas poslední zaznamenané aktivity
lastUpdated	DateTime	Datum a čas poslední zaznamenané změny
columns	String[]	Seznam sloupců v datech
data	DataJsonLD (C.6.7)	Uložená data ve formátu JSON-LD

C.3.3 DataColumn

Entita *DataColumn* představuje jeden sloupec, který přísluší k jednomu konkrétnímu zdroji dat. Jsou zde obsaženA nastavení zobrazení a chování tohoto sloupce nebo zpracovaná metadata.

Tabulka C.7: Popis entity DataColumn

Atribut	Typ	Popis
dataSourceId	ID*	ID zdroje dat
key	String*	Neměnný klíč sloupce odpovídající hodnotě v datech
dataType	DataType* (C.5.5)	Přiřazený datový typ sloupce
aggregations	ColumnAggregations (C.6.11)	Zpracované statistiky sloupce
queryPositions	ColumnQueryPositions (C.6.12)	Zpracované pozice sloupce z dotazu
properties	ColumnProperties (C.6.9)	Nastavení sloupce
samplePredicates	ColumnSamplePredicates (C.6.15)	Získaný vzorek vlastností
iriValues	ColumnIriValues (C.6.14)	Získané vlastnosti IRI hodnot
prefixView	Boolean	Zobrazení v prefixové formě

C.3.4 DataTransformation

Entita *DataTransformation* představuje jednu transformaci, která může být na zdroj dat aplikována. Atributy jsou použity dle typu transformacec.

Tabulka C.8: Popis entity DataTransformation

Atribut	Typ	Popis
dataSourceId	ID*	ID zdroje dat

(tabulka pokračuje na další stránce)

Tabulka C.8 *(pokračování z předchozí stránky)*

Atribut	Datový typ	Popis
type	DataTransformation-Type* (C.5.6)	Typ transformace
display	String	Text zobrazení transformace
iriPredicate	TransformationIri-Predicate (C.6.16)	Nastavení predikátu pro IRI transformace
filters	Record[String, Re-cord[String, ColumnFil-ter (C.6.17)]]	Nastavení filtrů dle zdroje dat a sloupce
columnKey	String	Klíč zpracovávaného sloupce

C.4 Dashboard

Pro dashboardy existuje aktuálně jedna základní entita *Dashboard*. Entita představuje jeden dashboard, včetně všech jeho stránek s vizuály a jejich nastaveními.

Tabulka C.9: Popis entity Dashboard

Atribut	Typ	Popis
id	ID*	ID dashboardu
userId	ID*	ID vlastníka
name	String*	Název dashboardu
dataSources	ID[]	Seznam použitých zdrojů dat
pages	DashboardPage[] (C.6.1)	Seznam stránek dashboardu
lastUsed	DateTime	Poslední aktivita
lastModified	DateTime	Poslední úprava
created	DateTime	Vytvořeno

C.5 Databázové číselníky

C.5.1 VisualType

Číselník rozlišuje typy vizuálů v dashboardu. Jeho hodnoty aktuálně mohou nabývat hodnot: *bar* (sloupcový graf), *pie* (koláčový graf), *doughnut* (koblihový graf), *line* (spojnicový graf), *radar* (polární diagram).

C.5.2 AggregationType

Číselník rozlišuje typy agregace používané u osy Y vizuálů dashboardu. Jeho hodnoty aktuálně mohou nabývat hodnot: *count* (počet), *average* (průměr), *sum* (suma), *max* (maximální hodnota), *min* (minimální hodnota), *firstValue* (první hodnota), *lastValue* (poslední hodnota).

C.5.3 SourceType

Číselník rozlišuje typ zdroje dat. Jeho hodnoty aktuálně mohou nabývat hodnot: *file* (soubor), *sparql* (SPARQL dotaz).

C.5.4 AuthorizationType

Číselník rozlišuje typ autorizace k SPARQL endpintu. Jeho hodnoty aktuálně mohou nabývat hodnot podporovaných způsobů autorizace: *BASIC*, *DIGEST*.

C.5.5 DataType

Číselník rozlišuje datový typ dle rozdělení v aplikaci. Jeho hodnoty aktuálně mohou nabývat hodnot: *iri* (IRI), *string* (řetězec), *number* (číslo), *boolean* (boolean), *dateTime* (datum a čas) a *unkown* (neznámý datový typ).

C.5.6 DataTransformationType

Číselník rozlišuje typy transformace dat. Jeho hodnoty aktuálně mohou nabývat hodnot: *addColumnFromIri* (přidání sloupce z hodnoty IRI sloupce), *addIriPredicate* (přidání hodnoty do IRI sloupce), *filter* (filtrace hodnot), *removeColumn* (odstranění sloupce) a *copyColumn* (kopírování sloupce).

C.6 Databázové objekty

C.6.1 DashboardPage

Objekt *DashboardPage* představuje jednu stránku v rámci Dashboardu.

Tabulka C.10: Popis objektu DashboardPage

Atribut	Typ	Popis
name	String*	Název stránky
visuals	DashboardVisual[] (C.6.2)	Seznam vizuálů
filters	DashboardFilter[] (C.6.4)	Seznam nastavení filtrů
sort	DashboardSort[] (C.6.5)	Seznam nastavení řazení

C.6.2 DashboardVisual

Objekt *DashboardVisual* představuje jeden vizuál a jeho nastavení v rámci stránky Dashboardu.

Tabulka C.11: Popis objektu DashboardVisual

Atribut	Typ	Popis
type	VisualType* (C.5.1)	Typ vizuálu
height	Float*	Výška vizuálu
width	Float*	Šířka vizuálu
x	Float*	Horizontální umístění vizuálu
y	Float*	Vertikální umístění vizuálu
title	String	Název grafu
xAxisLabel	String	Popisek osy X
yAxisLabel	String	Popisek osy Y
dataSourceId	ID	Přiřazený zdroj dat
labels	String[]	Seznam klíčů kategorických proměnných osy X
values	VisualValue[] (C.6.3)	Nastavení proměnných osy Y

C.6.3 VisualValue

Objekt *VisualValue* představuje nastavení proměnných vizuálu a jejich agregací na ose Y.

Tabulka C.12: Popis objektu VisualValue

Atribut	Typ	Popis
column	String*	Klíč agregovaného sloupce
aggregation	AggregationType* (C.5.2)	Typ agregace

C.6.4 DashboardFilter

Objekt představuje nastavení filtrů jednoho zdroje dat v dashboardu.

Tabulka C.13: Popis objektu DashboardFilter

Atribut	Typ	Popis
dataSourceId	ID*	Klíč zdroje dat
column	String*	Klíč filtrovaného sloupce
order	ColumnFilter* C.6.17	Nastavení filtrace

C.6.5 DashboardSort

Objekt představuje nastavení řazení jednoho zdroje dat v dashboardu.

Tabulka C.14: Popis objektu DashboardSort

Atribut	Typ	Popis
dataSourceId	ID*	Klíč zdroje dat
column	String*	Klíč řazeného sloupce
order	String*	Typ řazení (asc, desc)

C.6.6 SourceObject

Objekt představuje nastavení zdroje dat. Jeho atributy se liší dle typu zdroje dat. Tabulka C.24 popisuje podobu entity v případě SPARQL zdroje dat. Další podoby (např. souborový zdroj dat) aktuálně nejsou navrženy.

Tabulka C.15: Popis objektu SPARQLObject

Atribut	Typ	Popis
url	String*	URL SPARQL Endpointu
query	String*	SPARQL dotaz
authorizationType	AuthorizationType (C.5.4)	Typ autorizace
username	String	Přihlašovací jméno
password	String	Heslo

C.6.7 DataJsonLD

Objekt *DataJsonLD* představuje data ve formátu JSON-LD viz E.

C.6.8 EntryJsonLD

Objekt *EntryJsonLD* představuje jeden záznam z řádky (*results, bindings* viz E).

C.6.9 ColumnProperties

Objekt *ColumnProperties* představuje konkrétní nastavení zobrazení daného sloupce.

Tabulka C.16: Popis objektu ColumnProperties

Atribut	Typ	Popis
columnDisplay	String	Text zobrazeného názvu sloupce nebo klíč názvu z propertiesObject
columnDescription	String	Text popisu sloupce nebo klíč popisu z *propertiesObject*

(tabulka pokračuje na další stránce)

Tabulka C.16 (*pokračování z předchozí stránky*)

Atribut	Datový typ	Popis
propertiesObject	Dict[String, EntryJsonLD (C.6.8)]	Získané vlastnosti sloupce z pozice proměnné
displayObject	ColumnDisplay (C.6.10)>	Nastavení zobrazení sloupce

C.6.10 ColumnDisplay

Objekt *ColumnDisplay* představuje nastavení zobrazení sloupce včetně specifických nastavení dle datového typu sloupce.

Tabulka C.17: Popis objektu ColumnDisplay

Atribut	Typ	Popis
columnWidth	Float	Šířka sloupce
align	String	Zarovnání sloupce (*left, right, center*)
inputDatePattern	String	Vzor zpracování datumu
outputDatePattern	String	Vzor zobrazení datumu
monthDisplay	String	Zobrazení měsíce (number, czech, english)
trueDisplay	String	Zobrazení hodnoty true
falseDisplay	String	Zobrazení hodnoty false
afterDecimalNums	Integer	Počet zobrazených desetinných čísel
decimalSeparator	String	Znak oddělení desetinné části
digitsGroupBy	Integer	Počet seskupovaných číslic
groupSeparator	String	Znak oddělení seskupovaných číslic

C.6.11 ColumnAggregations

Objekt *ColumnAggregations* představuje vypočítané statistiky sloupce.

Tabulka C.18: Popis objektu ColumnAggregations

Atribut	Typ	Popis
rowsCounted	Integer	Celkový počet řádek
processingDuration	Float	Doba výpočtu statistik
nullCount	Integer	Počet řádek bez výskytu sloupce
iriCount	Integer	Počet řádek IRI
literalCount	Integer	Počet řádek literálu
bnodeCount	Integer	Počet řádek bnode
perValueCount	Dict[String, Integer]	Počty hodnot literálů
perDatatypeCount	Dict[String, Integer]	Počty datových typů literálů
perClassCount	Dict[String, Integer]	Počty IRI hodnot
perLangCount	Dict[String, Integer]	Počty jazyků

C.6.12 ColumnQueryPositions

Objekt *ColumnQueryPositions* představuje zpracované pozice sloupce ze SPARQL dotazu a informace o nalezených proměnných v ostatních pozicích.

Tabulka C.19: Popis objektu ColumnQueryPositions

Atribut	Typ	Popis
isSubjectWith	Dict[String, Triple (C.6.13)]	Nalezené trojice v pozici subjekt
isPredicateOf	Dict[String, Triple (C.6.13)]	Nalezené trojice v pozici predikát
isObjectOf	Dict[String, Triple (C.6.13)]	Nalezené trojice v pozici objekt

C.6.13 Triple

Objekt *Triple* představuje jednu zpracovánou trojici ze SPARQL dotazu.

Tabulka C.20: Popis objektu Triple

Atribut	Typ	Popis
subject	String	Hodnota na pozici subjektu
predicate	String	Hodnota na pozici predikátu
object	String	Hodnota na pozici objektu

C.6.14 ColumnIriValues

Objekt *ColumnIriValues* uchovává data získaných alternativních vlastnosti k IRI hodnotám ve sloupci.

Tabulka C.21: Popis objektu ColumnIriValues

Atribut	Typ	Popis
display	String	Klíč načtené hodnoty, která má být zobrazena místo IRI
loadedPredicates	String[]	Seznam získaných vlastností
values	Record[String, Record[String, EntryJsonLD (C.6.8)]]	Načtené alternativní hodnoty dle IRI hodnoty a získané vlastnosti

C.6.15 ColumnSamplePredicates

Objekt *ColumnSamplePredicates* uchovává data získaných alternativních vlastnosti k IRI hodnotám ve sloupci.

Tabulka C.22: Popis objektu ColumnSamplePredicates

Atribut	Typ	Popis
sampleIris	String[]	Seznam vzorků IRI, ze kterých byly získány hodnoty
values	Record[String, EntryJsonLD (C.6.8)]	Načtené vlastnosti vzorků IRI hodnot

C.6.16 TransformationIriPredicate

Objekt *TransformationIriPredicate* slouží k nastavení transformace pracující s IRI sloupci.

Tabulka C.23: Popis objektu TransformationIriPredicate

Atribut	Typ	Popis
iriColumn	String*	Klíč zdrojového IRI sloupce
predicate	String*	Vlastnost zdrojového IRI sloupce
newColumnName	String	Název nového sloupce

C.6.17 ColumnFilter

Objekt *ColumnFilter* představuje nastavení filtrace jednoho sloupce.

Tabulka C.24: Popis objektu ColumnFilter

Atribut	Typ	Popis
type	Dict[String, Boolean]	Filtrace dle typu
datatype	Dict[String, Boolean]	Filtrace dle datového typu literálů
value	Dict[String, Boolean]	Filtrace dle hodnoty literálu
class	Dict[String, Boolean]	Filtrace dle IRI třídy
xml:lang	Dict[String, Boolean]	Filtrace dle jazyka

Validační dotazy D

Validační dotazy byly poskytnuty vedoucím práce a pracují s endpointem *mre.zcu.cz* s neveřejným přístupem. Jsou doplněny o dotaz z endpointu *dbpedia*, aby bylo ověřeno, že aplikace dokáže pracovat i s jinými endpointy. Dotazy pokrývají všechny podporované datové typy a rozsah většího i menšího množství dat. Samotné zdroje dotazů je možné nalézt v elektronické příloze popsané v části A.

D.1 Dotaz 1 - CT vyšetření

Dotaz pracující s endpointem *mre.zcu.cz* vrací data o CT vyšetřeních. Obsahuje identifikátory vyšetření, booleovskou hodnotu indikující, zda bylo vyšetření provedeno, případně datum a čas daného vyšetření. Informace se opakují pro další fáze CT vyšetření, jako je příjem po 24 hodinách a 7 dnech. Dotaz je vhodný pro prozkoumání zpracování sloupců literálů, kdy se zde nachází řetězec, boolean nebo datum a čas. Dotaz při validačním průzkumu vrací 5879 řádků a 14 sloupců.

D.2 Dotaz 2 - Akutní intervence

Data získaná z dotazu pracujícího s endpointem *mre.zcu.cz* představují záznamy o akutních intervencích (reakce na případ cévní mozkové mrtvice). Obsahují identifikátory případu, booleovskou hodnotu, zda bylo provedena trombolýza (rozpuštění krevní sraženiny), datum a čas zahájení trombolýzy. Dále jsou zde informace o lokalizaci jednotlivých sledovaných hodnot. Množství hodnot je ve formátu IRI a dotaz je vhodný pro prozkoumání zpracování sloupců IRI. Dotaz při validačním průzkumu vrací 5815 řádků a 22 sloupců, z toho 12 IRI sloupců.

D.3 Dotaz 3 - Populace měst

Data dotazu z endpointu *dbpedia.org* obsahují populaci měst a hodnotu jejich odpovídající země. Obsahují IRI hodnotu města, země a číselnou hodnotu počtu obyvatel. Tento dotaz je tedy vhodný pro prozkoumání zpracování sloupců IRI na úložišti

dbpedia a zpracování číselného literálu. Dotaz vrací 10 000 řádků a 3 sloupce (z toho 2 IRI sloupce). Jedná se o aplikací neřešené omezení SPARQL Endpointu dbpedia, které vrací maximálně 10 000 záznamů.

D.4 Dotaz 4 - Přehled obrazových sérií

Dotaz z endpointu *mre.zcu.cz* obsahuje data o sériích obrazových snímků. Obsahuje identifikátor pacienta, název studie, identifikátor studie, název obrazové série, číslo série a počet souborů obrazové série. Dotaz při validačním průzkumu vrací 129 367 řádků a 6 sloupců literálů. Dotaz je tedy vhodný pro ověření zpracování velkého množství dat.

Ukázka odpovědi JSON-LD

E.1 ASK dotaz

```
{
  "head" : { } ,
  "boolean" : true
}
```

Zdrojový kód E.1: Ukázka odpovědi ASK dotazu

E.2 SELECT dotaz

```
"head": {
    "vars": [ "book" , "title" ]
    "link": [ "http://ex.org/dataset/metadata.ttl" ]
},
"results": {
    "bindings": [
      {
        // blank node with label R1
        "book": { "type": "bnode" , "value": "r1" } ,
        "title": {
            "type": "literal",
            "value": "Harry Potter and the Half-Blood Prince"
        }
      },
      {
        "book": {
            "type": "uri",
            "value": "http://ex.org/book/book7"
        },
        "title": {
            "type": "literal",
    "datatype": "http://www.w3.org/2001/XMLSchema#string ,
```

```
23      "xml:lang": "en", // language tag
24              "value": "Harry Potter and the Deathly Hallows"
25          }
26        }
27       ]
28    }
29 }
```

Zdrojový kód E.2: Ukázka odpovědi SELECT dotazu

Ukázka implementovaných zobrazení

F

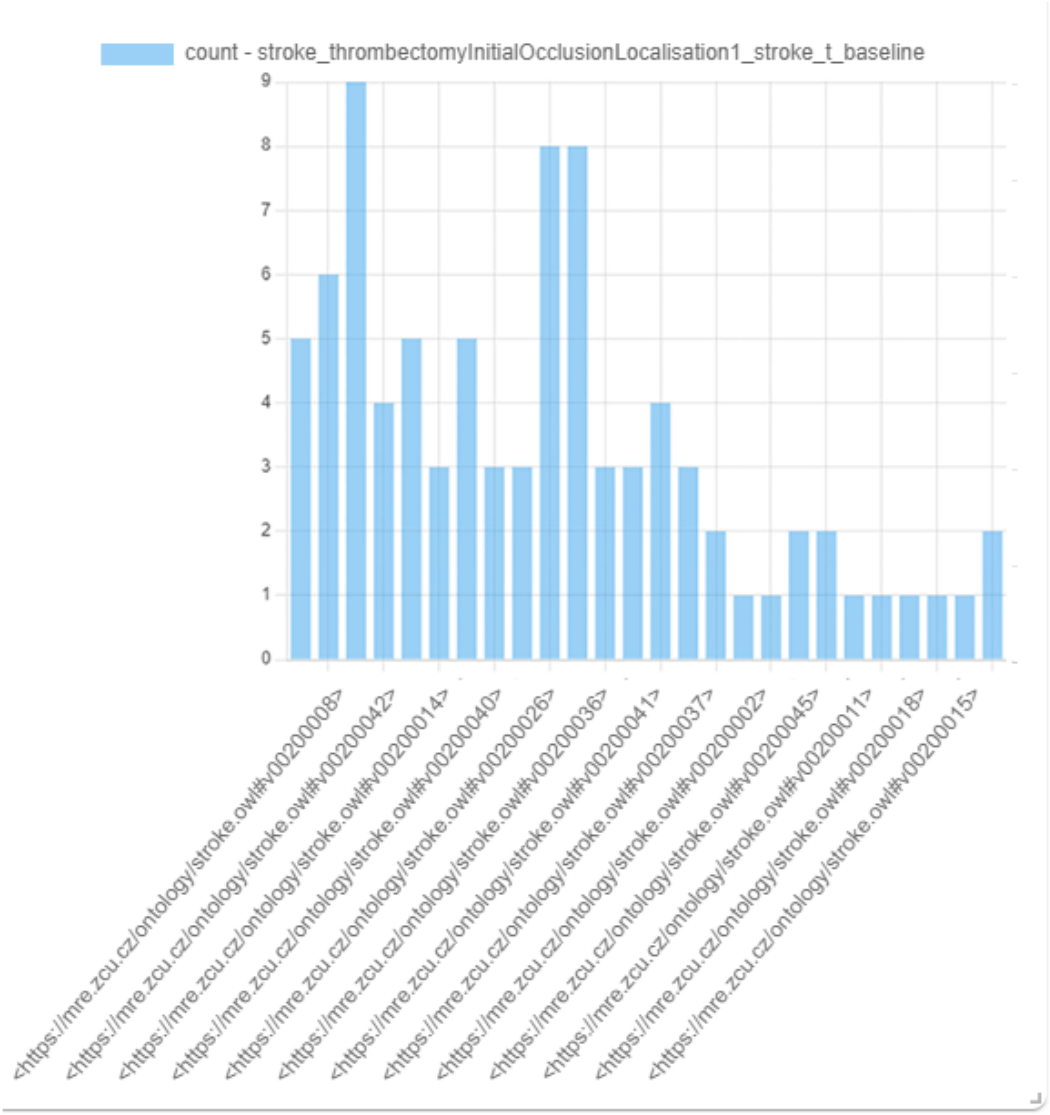
< BACK | DBPEDIA cities countries | SOURCE | DATA | TRANSFORMATIONS | Remove

iri city	iri country	123 population
Jiran(en)	India(en)	1
Pembroke, Kentucky(en)	United States(en)	2
Gulabpura(en)	India(en)	3
Maza, North Dakota(en)	United States(en)	5
Athelstan, Iowa(en)	United States(en)	6
Clayton, Idaho(en)	United States(en)	7
Gangaghat(en)	India(en)	9
Beaconsfield, Iowa(en)	United States(en)	15
Conway, Iowa(en)	United States(en)	17
Ellston, Iowa(en)	United States(en)	19
Durango, Iowa(en)	United States(en)	20
Maloy, Iowa(en)	United States(en)	22
Berkley, Iowa(en)	United States(en)	23
Bettles, Alaska(en)	United States(en)	23
Bankston, Iowa(en)	United States(en)	23
Edna Bay, Alaska(en)	United States(en)	25
Delphos, Iowa(en)	United States(en)	26
Coburg, Iowa(en)	United States(en)	26
Galt, Iowa(en)	United States(en)	26
Hepburn, Iowa(en)	United States(en)	26
Jolley, Iowa(en)	United States(en)	28
Udell, Iowa(en)	United States(en)	28
Bay Lake, Florida(en)	United States(en)	29
Blanchard, Iowa(en)	United States(en)	29
Elkport, Iowa(en)	United States(en)	29
Colona, Colorado(en)	United States(en)	30
Gillett Grove, Iowa(en)	United States(en)	30
Rodman, Iowa(en)	United States(en)	31
Mount Sterling, Iowa(en)	United States(en)	33
Aspinwall, Iowa(en)	United States(en)	33
Buck Grove, Iowa(en)	United States(en)	34
أرلسا(ar)	United States(en)	34
Carbon, Iowa(en)	United States(en)	36
Coppock, Iowa(en)	United States(en)	36
Attirala(en)	India(en)	36
Curlew, Iowa(en)	United States(en)	37
Jackson Junction, Iowa(en)	United States(en)	37
Rinard, Iowa(en)	United States(en)	38

COPY FILE URL | DOWNLOAD

Displaying 200 out of 4374 rows. LOAD MORE | LOAD ALL

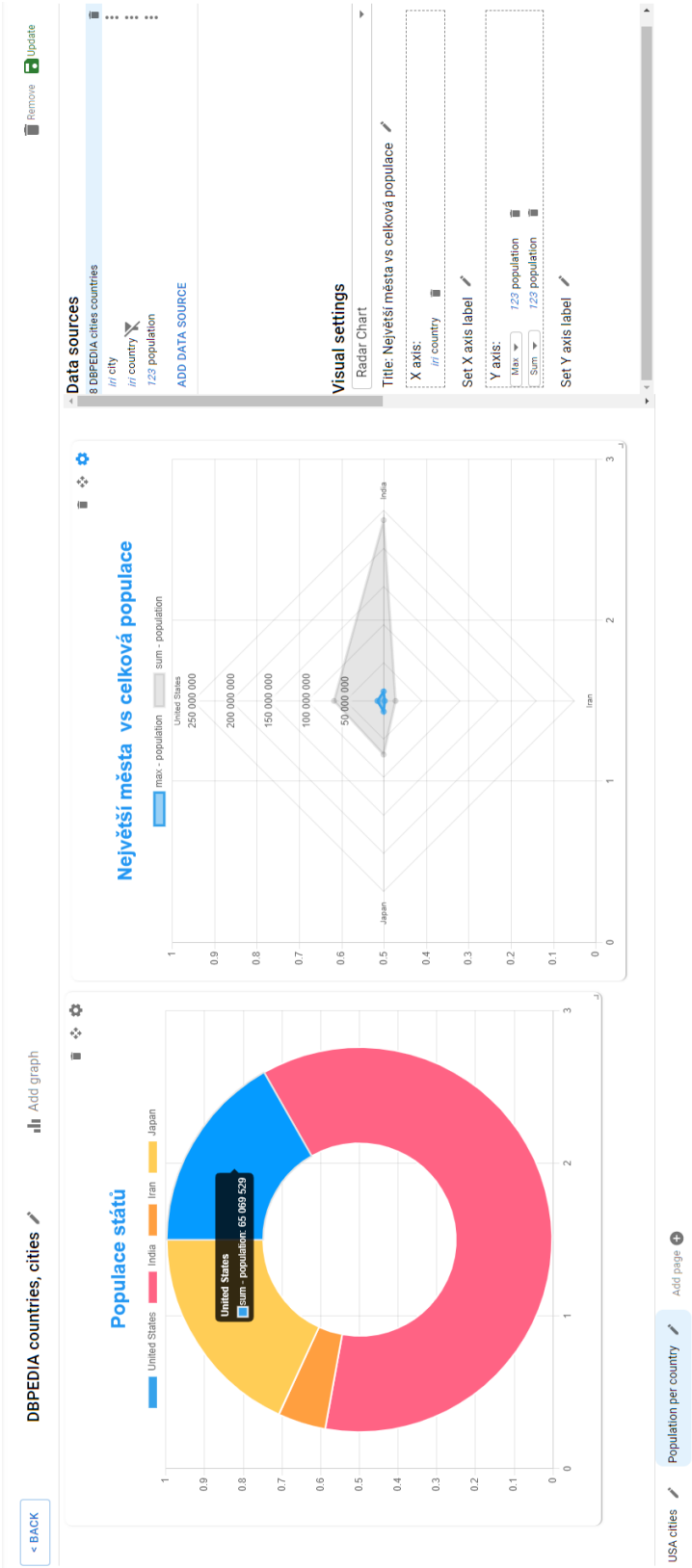
Obrázek F.1: Ukázka zobrazení přehledu dat



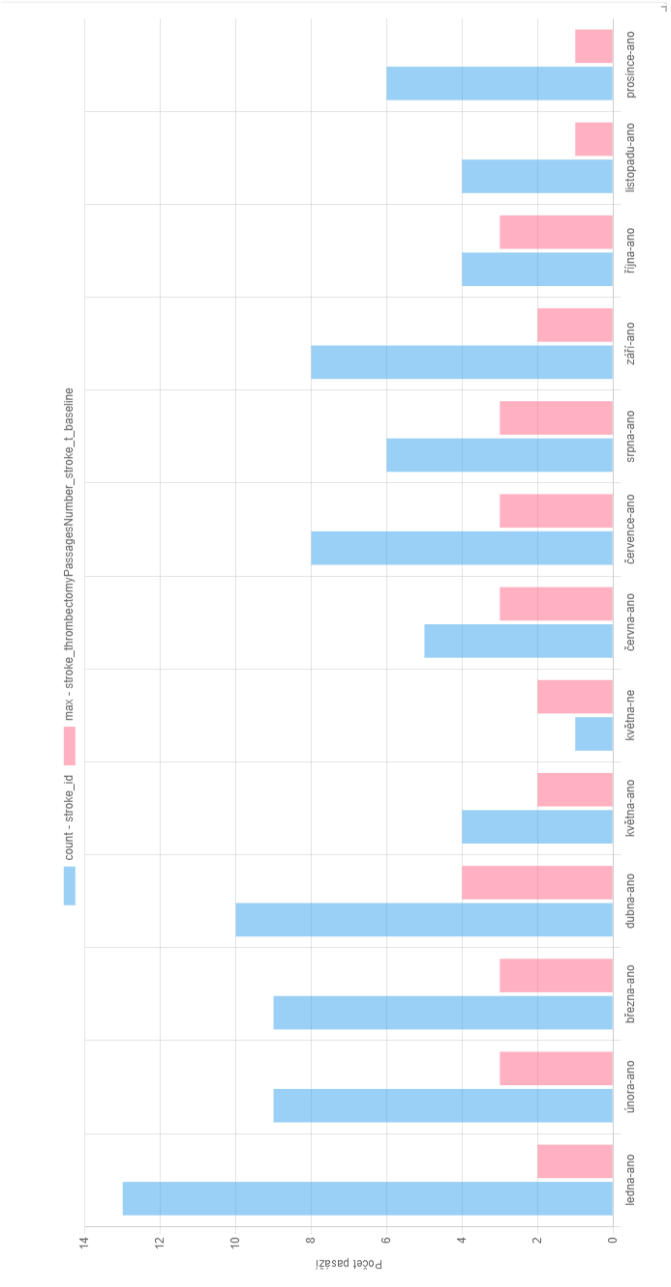
Obrázek F.2: Ukázka vizuálu s klasickými IRI hodnotami

Obrázek F.3: Ukázka vizuálu s implementovaným IRI zobrazením

Obrázek F.4: Ukázka implementovaného IRI detailu



Obrázek F.5: Ukázka implementované obrazovky dashboardů



Obrázek F.6: Ukázka vizuálu s implementovanými zobrazeními literálů

Uživatelská dokumentace

G

G.1 Sestavení

G.1.1 Konfigurace databáze a RDF úložiště

Přes sestavením aplikace je nutná konfigurace připojení k aplikační databázi MongoDB a volitelná konfigurace připojení k RDF úložišti slovníků a ontologií. Pokud není připojení k RDF úložišti správně nastaveno, nebude používáno.

Návod na konfiguraci databáze Mongo DB:

- V serverovém adresáři aplikace */backend_flask* otevřete */shared/database.py*.
- Do proměnné `mongoUri` nastavte řetězec připojení k Mongo DB.
- Do proměnné `dbName` vložte název databáze.

Návod na konfiguraci RDF úložiště:

- V serverovém adresáři aplikace */backend_flask* otevřete */services/sparql/RDFStorageService.py*.
- Do proměnných třídy `RDFSotrageService` vyplňte údaje k připojení: `endpointUrl` (URL), `authorizationType` (typ autorizace: *None*, *"Digest auth"*, *"Basic auth"*), `username` (autorizační jméno), `password` (autorizační heslo), `graph` (*uri* podgrafu v úložišti).

G.1.2 Sestavení aplikace pro nasazení

Návod k sestavení serveru i klienta aplikace v prostředí Docker:

1. Stáhněte a nainstalujte aplikaci *Docker Compose*, která je dostupná na stránkách https://docs.docker.com/compose/install/.

2. Pro sestavení kontejnérů serverové i backendové části aplikace spusťte terminál v kořenovém adresáři se zdrojovými kódy (A) a proveďte příkaz *docker-compose up*. Oba kontejnery se sestaví a spustí.

3. Další možnosti práce se sestavenými kontejnery včetně jejich sdílení je možné zjistit v dokumentaci, která je dostupná na stránkách https://docs.docker.com/.

G.1.3 Sestavení klienta pro vývoj

Návod k sestavení klienta aplikace:

1. Stáhněte a nainstalujte aplikace *Node.js* a *NPM*, které jsou dostupné na stránkách https://nodejs.org/en/download.

2. Pro instalaci knihoven spusťte terminál v kořenovém adresáři klienta (A) a proveďte příkaz *npm install*.

3. Spuštění klienta po instalaci umožní příkaz *npm start*.

G.1.4 Sestavení serveru pro vývoj

Návod k sestavení serveru aplikace:

1. Stáhněte a nainstalujte aplikaci *Python 3.9 nebo vyšší verzi*, která je dostupný na stránkách https://www.python.org/downloads/.

2. Pro instalaci knihoven spusťte terminál v kořenovém adresáři serveru (A) a proveďte příkaz *pip install -r requirements.txt*.

3. Spuštění serveru po instalaci umožní příkaz *python main.py*.

G.2 Ovládání aplikace

V této části bude popsáno ovládání jednotlivých obrazovek a komponent aplikace. Zobrazení některých ukázek je pro možnost použití v textu značně přiblíženo. Při použití webové aplikace by bylo rozložení prvků na obrazovce více rozprostřené.

G.2.1 Navigační lišta

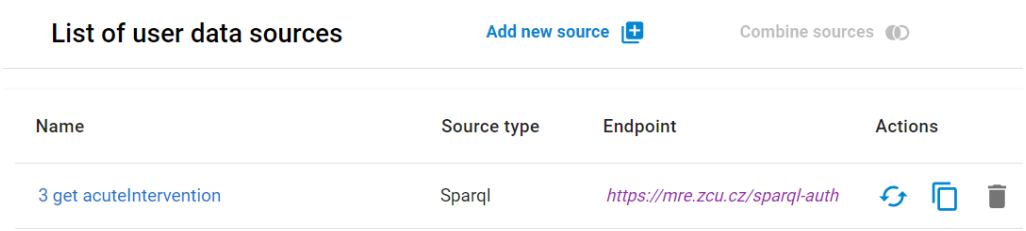
Pro pohyb mezi jednotlivými obrazovkami aplikace použijte navigační lištu (obrázek G.1), která se nachází nad obsahem každé z obrazovek. Pro přesměrování na domovskou obrazovku aplikace klikněte na logo, nebo název aplikace. K přesměrování na obrazovku zdrojů dat (G.2.2) poslouží tlačítko *DATA SOURCES*. Tlačítko *DASHBOARDS* slouží k přesměrování na obrazovku dashboardů (G.2.3 a *SPARQL*

SETTINGS na obrazovku SPARQL nastavení (G.2.4). V případě, že není uživatel přihlášen, nachází se na pravém konci navigační lišty přesměrování na obrazovku registrace (*Register*) a přihlášení (*Login*). V opačném případě je zde zobrazeno jméno prihlášeného uživatele s možností odhlášení.

Obrázek G.1: Navigační lišta aplikace

G.2.2 Zdroje dat

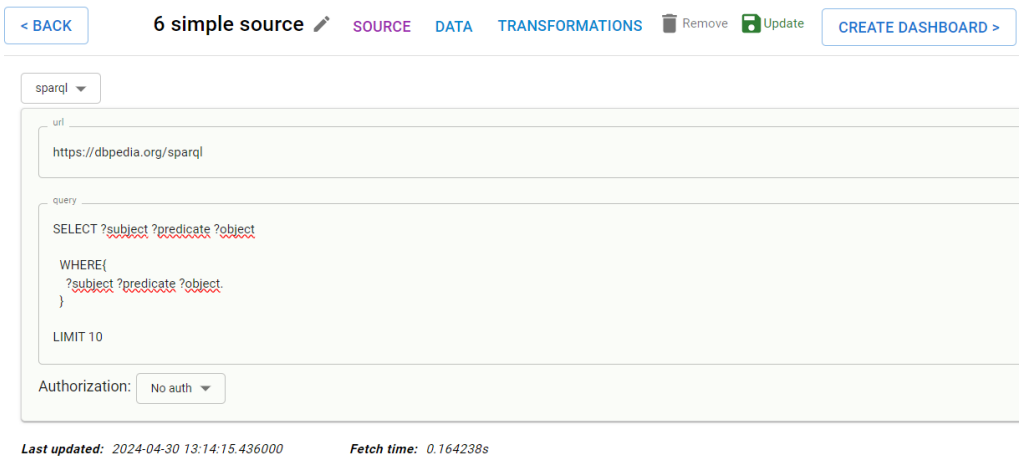
Po přesměrování do obrazovky zdrojů dat se před vámi vyskytuje seznam vámi vytvořených zdrojů dat (obrázek G.2). Na horní liště je tlačítko přidání nového zdroje (*Add new source*), které vás přesměruje na obrazovku vytvoření nového zdroje dat. Kliknutím na název zdroje v seznamu budete na tento zdroj přesměrováni. Kliknutí na endpoint daného zdroje otevře adresu v novém okně. Nakonec jsou zde tlačítka, které budou popsána v pořadí zleva: aktualizace zdroje (vynutí nové vykonání SPARQL dotazu nad zdrojem), kopírování zdroje (zkopíruje zdroj dat), mazání (po konfirmaci smaže zdroj dat).



Obrázek G.2: Seznam zdrojů dat

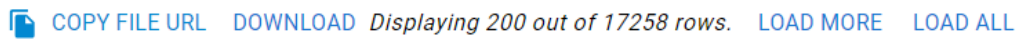
Detail zdroje dat (obrázek G.3) pak obsahuje horní lištu, která nejprve obsahuje tlačítko pro návrat do sezmamu zdrojů dat (*BACK*) a název zdroje dat (*kliknutím na tlačítko editace je možné název měnit*). Lišta pokračuje tlačítky pro přepínání mezi pohledem na nastavení zdroje dat (*SOURCE*), data (*DATA*) a pohledem s transformacemi (*TRANSFORMATIONS*). Na pravé straně lišty jsou pak k dispozici tlačítka na uložení nastavení zdroje (v případě změny zdroje dat aktualizace) a smazání zdroje.

Na obrázku G.3 je možné vidět zvolený pohled na nastavení zdroje dat. V nastavení je vybrán *sparql* zdroj dat, kterému přísluší nastavení endpointu (*url*) a dotazu (*query*). Není nastavena žádná autorizace (*No auth*) a nejsou tedy zobrazena pole pro nastavení příhlašovacích údajů. U každého pohledu je v případě, že zbývá prostor na obrazovce, zobrazena i tabulka s daty (obrázek G.5).



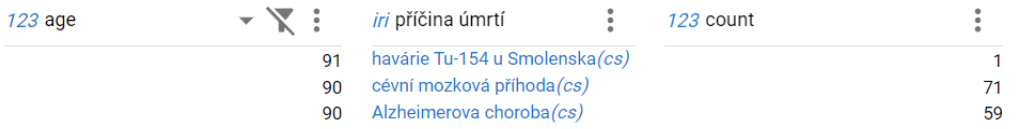
Obrázek G.3: Nastavení zdroje dat

Na spodní části obrazovky zdroje dat se pak vždy nachází lišta, která je zobrazena na obrázku G.4. Umožňuje zkopírování adresy ke stažení dat (*COPY FILE URL*), tlačítko stažení dat (*DOWNLOAD*) a dále zobrazení více dat (*LOAD MORE*) nebo dokonce všech dat (*LOAD ALL*).



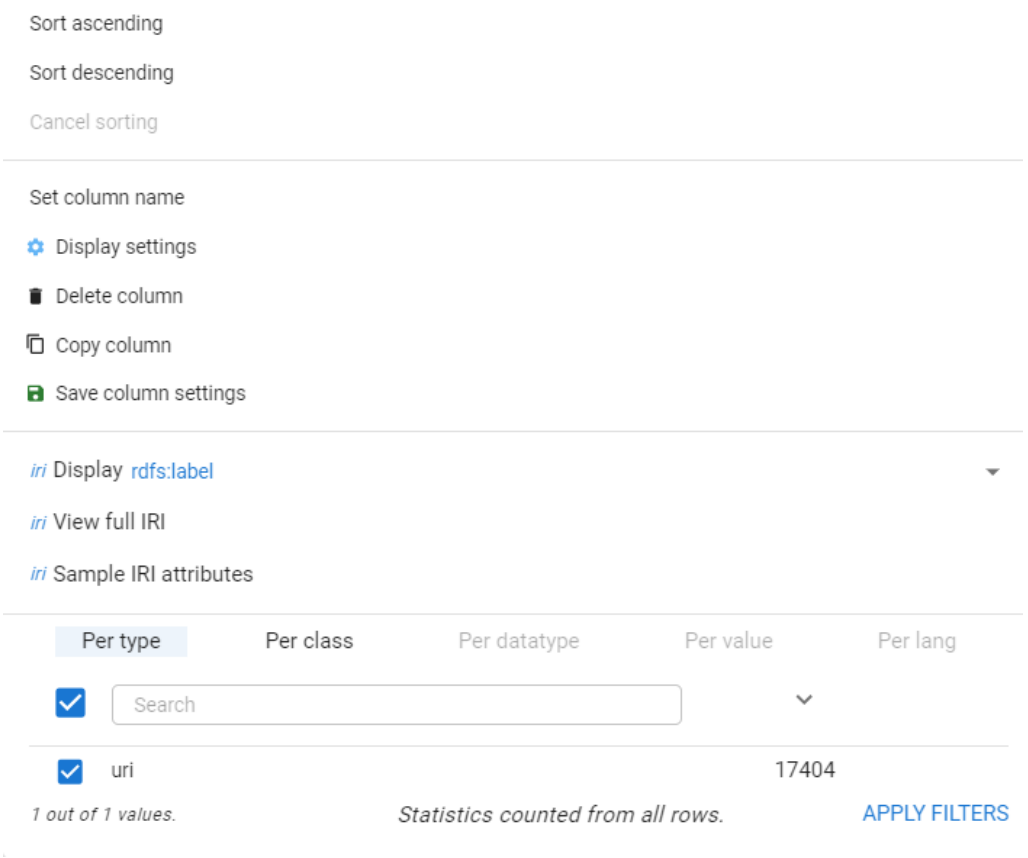
Obrázek G.4: Spodní lišta zdroje dat

Samotná tabulka pak zobrazuje data po jednotlivých sloupcích, jak ukazuje obrázek G.5. Každý sloupec má ve smém záhlaví přiřazený datový typ, název a v případě, že je nastaveno řazení, i odpovídající tlačítko řazení (u sloupce *age*). Po kliknutí na tlačítko řazení je sloupec seřazen obráceným směrem. Následuje tlačítko zrušení filtrů, které je opět zobrazeno pouze v případě, že jsou ve sloupci aktivní filtry. Posledním tlačítkem v záhlaví je tlačítko nastavení zvoleného sloupce, které otevře menu na obrázku G.6.



Obrázek G.5: Zobrazení získaných dat

Menu na obrázku G.6 obsahuje řazení a nastavení názvu sloupce pro každý datový typ sloupce. Dále je zde nastavení zobrazení sloupce (obrázek G.7), tlačítko smazání nebo zkopírování sloupce a naposledy tlačítko pro uložení změn v nastavení sloupce. Pokud se jedná o IRI sloupec, je k dispozici ještě volba zobrazení hodnot

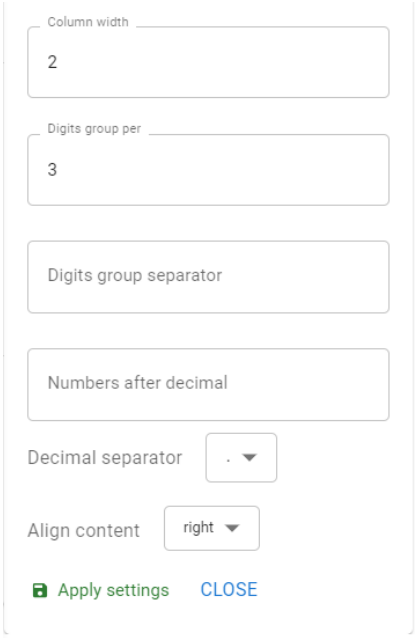


Obrázek G.6: Nastavení sloupce

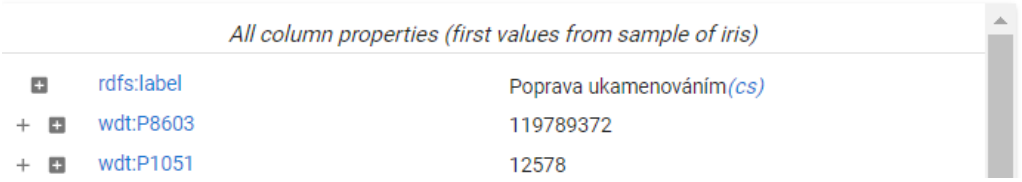
ve sloupci, volba prefixové nebo úplné IRI formy a vzorek vlastností IRI hodnot ve sloupci (obrázek G.8). Nakonec každé menu sloupce obsahuje filtrační tabulku, která rozlišuje všechny hodnoty ve sloupci dle typu (Per type), IRI hodnoty (Per class), datového typu literálu (Per datatype), hodnoty literálu (Per value) a případného jazyka (Per language). Hodnoty lze dle zvolené kategorie vyhledávat a řadit. Po nastavení filtrů a stisknutí tlačítka *APPLY FILTERS* je filtrace nad daty provedena.

Menu pro nastavení zobrazení sloupce na obrázku G.7 se liší dle přiřazeného datového typu sloupce. Pro každý datový typ však umožňuje určit šířku sloupce (*Column width*) a zarovnání textu (*Align content*). Na zmíněném obrázku je možné vidět nastavení číselného zobrazení sloupce. Tlačítko *Apply settings* pak nastavení aplikuje na data a tlačítko *CLOSE* nastavení uzavře.

Obrázek G.8 zobrazuje menu se vzorem vlastností sloupce. Kliknutí na tlačítko bez ohraničení načte danou vlastnost do odpovídajícího sloupce a tmavé tlačítko pak přidá tuto vlastnost jako nový sloupec.



Obrázek G.7: Nastavení zobrazení sloupce

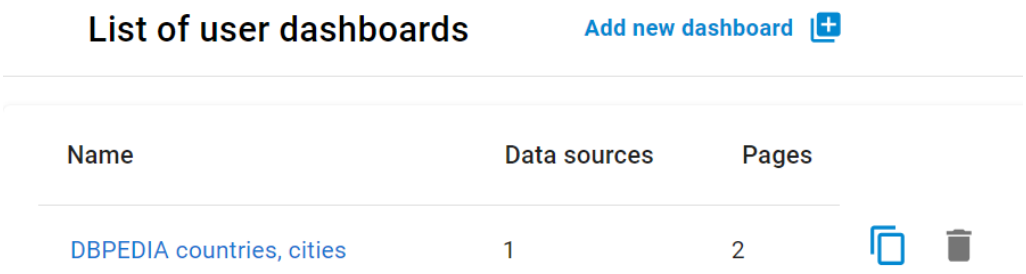


Obrázek G.8: Vzor IRI hodnot

G.2.3 Dashboardy

Po přesměrování do obrazovky zdrojů dashboardů se před vámi vyskytuje seznam dashboardů (obrázek G.9), které byly vámi vytvořeny. Na horní liště je tlačítko přidání nového dashboardu (*Add new dashboard*), které vás přesměruje na obrazovku vytvoření nového dashboardu. Kliknutím na název dashboardu v seznamu budete na tento dashboard přesměrováni. Nakonec jsou zde tlačítka, která budou popsána v pořadí zleva: kopírování dashboardu (zkopíruje dashboard), mazání (po konfirmaci smaže dashboard).

Detail dashboardu (obrázek G.10) má horní lištu, na níž je zobrazeno tlačítko pro návrat do sezmamu dashboardů (*BACK*) a název dashboardu (*kliknutím na tlačítko editace je možné název měnit*). Lišta pokračuje tlačítkem přidání grafu, které otevře seznam grafů, jež je možné přidat. Na pravé straně lišty jsou pak k dispozici tlačítka pro uložení nastavení dashboardu (v případě změny zdroje dat aktualizace) a smazání zdroje.



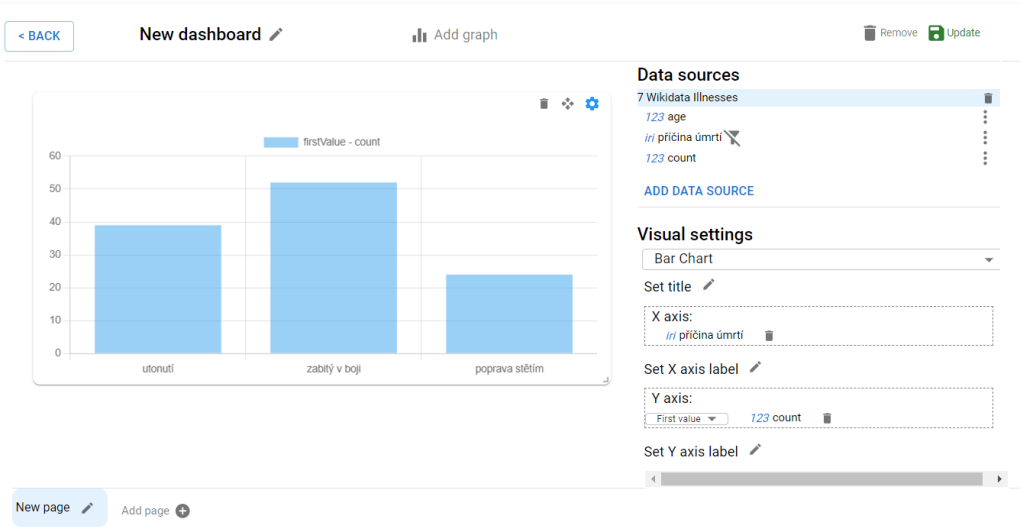
Obrázek G.9: Seznam dashboardů

Pod horní lištou se nachází zvolená stránka dashboardu a její vizuály. Vizuál má v pravém horním rohu tři tlačítka, kde první slouží k smazání vizuálu, druhé slouží k přemisťování vizuálu tažením a poslední k otevření nastavení dashboardu, které se objeví v pravém panelu. Pod obsahem stránky dashboardu se nachází spodní lišta, na které je možné přepínat a upravovat stránky dashboardu. Tlačítko editace slouží ke změně názvu stránky a *Add page* pak stránku přidává.

V pravém panelu obrazovky je nejprve zobrazen seznam přidaných zdrojů dat se svými sloupci, kde se zobrazení hlavičky sloupce chová stejně jako v obrazovce zdrojů dat. Tlačítkem *ADD DATA SOURCE* se otevře seznam s uživatelovými dashboardy, které je možné přidat. Pokud je aktivováno nastavení nějakého z vizuálů, nachází se pod seznamem zdrojů. První je zde výběrové menu pro volbu typu vizuálu a nastavení názvu vizuálu pomocí editační ikony u textu *Set title*. Do prostoru označeného názvem *X axis* lze myší přetáhnout hlavičky sloupců z datových zdrojů, které jsou pak nastaveny jako kategorické proměnné pro daný vizuál na ose X. U každého přidaného sloupce je tlačítko odebrání, které ho z nastavení odebere. Stejně se chová i prostor *Y axis*, kam se naopak přidávají sloupce, které budou na ose Y. Každý přidaný sloupec žádá výběrovým seznamem zvolení typu agregace. Každému vizuálu lze přiřadit libovolný počet proměnných osy X i Y, všechny však musí být ze stejného zdroje dat. Poslední možností je nastavení popisu os X (*Set X axis label*) a Y (*Set Y axis label*), které funguje stejně jako nastavení názvu grafu a slouží pro přiřazení popisku k osám.

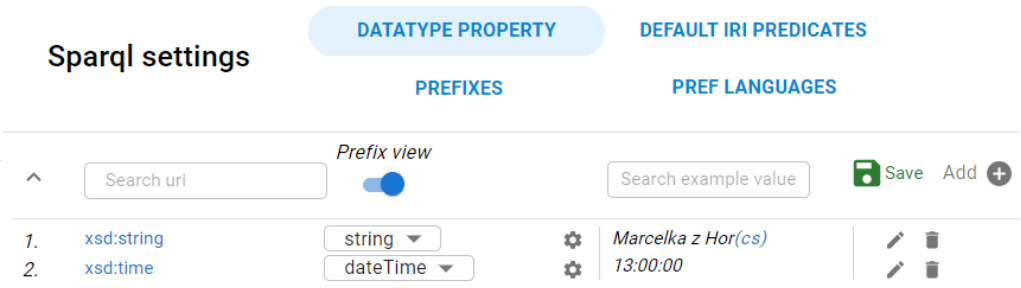
G.2.4 SPARQL nastavení

Po přesměrování do obrazovky SPARQL nastavení se před vámi vyskytuje právě zvolené nastavení (obrázek G.11 s výchozím zvolením nastavení datových typů). Na horní liště je tlačítko, které mezi jednotlivými nastaveními umožňuje přepínat. Konkrétně přísluší tlačítka následujícím nastavením: předvolby datových typů (*DATATYPE PROPERTY*), předvolby načítaných predikátů (*DEFAULT IRI PREDICATES*), předvolby prefixů (*PREFIXES*) a předvolby jazyků (*PREF LANGUAGES*).



Obrázek G.10: Nastavení dashboardu

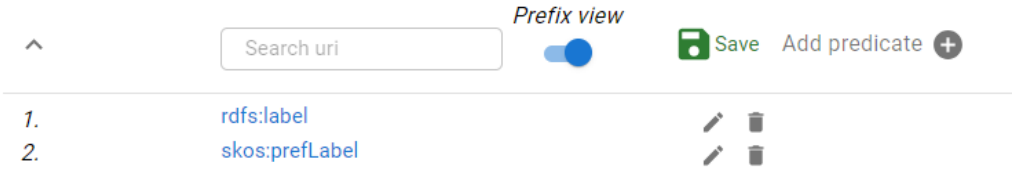
Obrazovka předvolby datových typů na obrázku G.11 obsahuje tabulku s nastavením datového typu dle uri třídy datového typu. Horní lišta tabulky umožňuje řazení dle posledního přidaného datového typu. Dále je zde možnost vyhledávání datového typu dle uri a zvolení zobrazení celého uri, nebo pouze jeho prefixové formy. V horní liště následuje možnost vyhledávání dle ukázkové hodnoty, a nakonec tlačítko uložení záznamů a přidání nového řádku. Samotný řádek pak na svém pravém konci obsahuje tlačítko smazání řádku a tlačítko editace, které umožňuje upravovat obsah uri hodnoty a ukázkové hodnoty. Výběrový seznam určuje namapovaný datový typ, podle kterého bude zobrazeno příslušné nastavení zobrazení sloupce po stisknutí tlačítka nastavení, jež otevře menu odpovídající obrázku G.6.



Obrázek G.11: Předvolby datových typů

Obrazovka předvolby datových typů na obrázku G.12 obsahuje tabulku s nastavením výchozích načítaných predikátů při zpracování IRI sloupců. Horní lišta tabulky umožňuje řazení dle posledního přidaného predikátu. Dále je zde možnost vyhledávání predikátu dle uri a zvolení zobrazení celého uri, nebo pouze jeho pre-

fixové formy. V horní liště následuje tlačítko uložení záznamů a přidání nového řádku. Samotný řádek pak na svém pravém konci obsahuje tlačítko smazání řádku a tlačítko editace, které umožňuje upravovat obsah uri hodnoty predikátu.

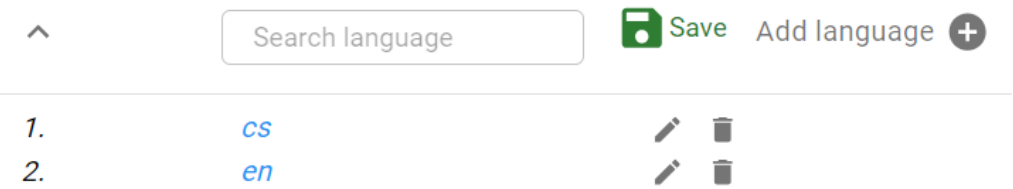


Obrázek G.12: Předvolby IRI vlastností

Obrazovka nastavení prefixů na obrázku G.13 obsahuje tabulku s prefixy, které jsou uživateli k dispozici při zobrazování IRI hodnot. Horní lišta tabulky umožňuje řazení dle posledního přidaného prefixu. Dále je zde možnost vyhledávání prefixu dle prefixu nebo celého uri. V horní liště následuje tlačítko uložení záznamů a přidání nového řádku. Samotný řádek na svém pravém konci obsahuje tlačítko smazání řádku a editace, které umožňuje upravovat obsah uri hodnoty i hodnoty prefixu.

Obrázek G.13: Předvolby prefixů

Obrazovka předvolby jazyků na obrázku G.14 obsahuje tabulku s nastavením výchozích jazyků uživatele, které se používají při získávání textových metadat sloupců. Horní lišta tabulky umožňuje řazení dle posledního přidaného jazyka. Dále je zde možnost vyhledávání jazyka. V horní liště následuje tlačítko uložení záznamů a přidání nového řádku. Samotný řádek pak na svém pravém konci obsahuje tlačítko smazání řádku a tlačítko editace, které umožňuje upravovat jazyk v daném řádku.



Obrázek G.14: Předvolby jazyků

Bibliografie

1. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 Concepts and Abstract Syntax* [online]. 2014-02-25. [cit. 2023-10-17]. Dostupné z: https://www.w3.org/TR/rdf11-concepts/.
2. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 Primer* [online]. 2014-06-24. [cit. 2023-10-15]. Dostupné z: https://www.w3.org/TR/rdf11-primer/.
3. WORLD WIDE WEB CONSORTIUM. *W3C Semantic Web Activity* [online]. 2001-11-02. [cit. 2023-10-15]. Dostupné z: https://www.w3.org/2001/12/semweb-fin/w3csw.
4. WORLD WIDE WEB CONSORTIUM. *RDF Schema 1.1* [online]. 2014-02-25. [cit. 2023-10-17]. Dostupné z: https://www.w3.org/TR/rdf-schema/.
5. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 N-Triples* [online]. 2014-02-25. [cit. 2023-10-18]. Dostupné z: https://www.w3.org/TR/n-triples/.
6. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 Turtle* [online]. 2014-02-25. [cit. 2023-10-18]. Dostupné z: https://www.w3.org/TR/turtle/.
7. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 TriG* [online]. 2014-02-25. [cit. 2023-10-18]. Dostupné z: https://www.w3.org/TR/trig/.
8. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 N-Quads* [online]. 2014-02-25. [cit. 2023-10-15]. Dostupné z: https://www.w3.org/TR/n-quads/.
9. WORLD WIDE WEB CONSORTIUM. *JSON-LD 1.1* [online]. 2020-07-16. [cit. 2023-10-19]. Dostupné z: https://www.w3.org/TR/json-ld11/.
10. WORLD WIDE WEB CONSORTIUM. *SPARQL 1.1 Query Results JSON Format* [online]. 2013-03-21. [cit. 2023-10-15]. Dostupné z: https://www.w3.org/TR/sparql11-results-json/.
11. WORLD WIDE WEB CONSORTIUM. *RDF 1.1 XML Syntax* [online]. 2014-02-25. [cit. 2023-10-21]. Dostupné z: https://www.w3.org/TR/rdf-syntax-grammar/.
12. WORLD WIDE WEB CONSORTIUM. *OWL Web Ontology Language* [online]. 2004-02-10. [cit. 2023-10-21]. Dostupné z: https://www.w3.org/TR/2004/REC-owl-features-20040210/.

13. GRUBER, Tom. Ontology. In: *Encyclopedia of Database Systems*. Ed. LIU, Ling; ÖZSU, M. Tamer. New York, NY: Springer New York, 2018, s. 2574–2576. ISBN 978-1-4614-8265-9. Dostupné z DOI: 10.1007/978-1-4614-8265-9_1318.

14. WORLD WIDE WEB CONSORTIUM. *SPARQL 1.1 Protocol* [online]. 2013-03-21. [cit. 2023-10-15]. Dostupné z: https://www.w3.org/TR/sparql11-protocol/.

15. OPENLINK SOFTWARE, INC. *OpenLink Virtuoso* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://virtuoso.openlinksw.com/.

16. THE APACHE SOFTWARE FOUNDATION. *Apache Jena Fuseki* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://jena.apache.org/documentation/fuseki2/.

17. ISLAM, Mohaiminul. Data Analysis: Types, Process, Methods, Techniques and Tools. *International Journal on Data Science and Technology*. 2020, roč. Vol. 6, No. 1, 2020, s. 10–15. Dostupné z DOI: 10.11648/j.ijdst.20200601.12.

18. SEENIVASAN, Dhamotharan. ETL (Extract, Transform, Load) Best Practices. *International Journal of Computer Trends and Technology*. 2023, roč. 71, s. 40–44. Dostupné z DOI: 10.14445/22312803/IJCTT-V71I1P106.

19. FRY, Ben. *Visualizing data: exploring and explaining data with the Processing environment*. 2008. ISBN 978-0-596-51455-6.

20. DETTORI, Joseph R; NORVELL, Daniel C. The Anatomy of Data. *Global Spine J*. 2018, roč. 8, č. 3, s. 311–313. Dostupné z DOI: 10.1177/2192568217746998.

21. BACH, Benjamin et al. *Dashboard Design Patterns*. 2022. Dostupné z arXiv: 2205.00757 [cs.HC].

22. MISHRA, P.; PANDEY, C. M.; SINGH, U.; KESHRI, A.; SABARETNAM, M. Selection of appropriate statistical methods for data analysis. *Annals of Cardiac Anaesthesia*. 2019, roč. 22, č. 3, s. 297–301. Dostupné z DOI: 10.4103/aca.ACA_248_18.

23. MICROSOFT CORPORATION. *Co je Power BI?* [online]. 2024-03-22. [cit. 2024-04-10]. Dostupné z: https://learn.microsoft.com/cs-cz/power-bi/fundamentals/power-bi-overview.

24. GOOGLE LLC IPA. *Looker Studio Overview* [online]. 2023. [cit. 2023-12-02]. Dostupné z: https://lookerstudio.google.com/overview.

25. SALESFORCE, INC. *Tableau Products* [online]. 2023. [cit. 2023-12-22]. Dostupné z: https://www.tableau.com/products.

26. SALESFORCE, INC. *What is Tableau?* [online]. 2023. [cit. 2023-12-22]. Dostupné z: https://www.tableau.com/why-tableau/what-is-tableau.

27. MICROSOFT CORPORATION. *Ceny Power BI* [online]. 2023. [cit. 2023-11-25]. Dostupné z: https://powerbi.microsoft.com/cs-cz/pricing/.

28. SALESFORCE, INC. *Tableau Pricing* [online]. 2023. [cit. 2023-12-22]. Dostupné z: https://www.tableau.com/pricing/teams-orgs.

29. HRUŠOVSKÝ, Jakub. *Zhodnocení nástrojů pro tvorbu grafů ve webovém prostředí: Comparison of web-based charting libraries*. Západočeská univerzita v Plzni, 2023. Dostupné také z: http://hdl.handle.net/11025/53925. bakalářská práce. Vedoucí práce Ing. KRYL Martin.

30. GOOGLE LLC IPA. *Google Charts* [online]. 2024. [cit. 2024-04-22]. Dostupné z: https://developers.google.com/chart.

31. MIKE BOSTOCK AND OBSERVABLE, INC. *D3* [online]. 2024. [cit. 2024-04-22]. Dostupné z: https://d3js.org/.

32. MONGODB, INC. *What is MongoDB?* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://www.mongodb.com/company/what-is-mongodb.

33. PYTHON SOFTWARE FOUNDATION. *What is Python? Executive Summary* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://www.python.org/doc/essays/blurb/.

34. SPARQL WRAPPER. *Sparql Wrapper* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://sparqlwrapper.readthedocs.io/en/latest/.

35. PALLETS. *Flask* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/.

36. MONGO ENGINE. *MongoEngine User Documentation* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://docs.mongoengine.org/.

37. REACT. *React Reference Overview* [online]. [cit. 2024-03-22]. Dostupné z: https://react.dev/reference/react.

38. MATERIAL UI SAS. *Material UI* [online]. 2024. [cit. 2024-03-22]. Dostupné z: https://mui.com/material-ui/.

39. TANSTACK. *TanStack Query* [online]. [cit. 2024-03-22]. Dostupné z: https://tanstack.com/query/v3.

40. PYTEST. *pytest* [online]. 2015. [cit. 2024-04-03]. Dostupné z: https://docs.pytest.org/en/8.1.x/.

41. VCELAK, Petr; KRYL, Martin; KLECKOVA, Jana. SPARQL Query-Builder for Medical Temporal Data. In: *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. 2018, s. 1–9. Dostupné z DOI: 10.1109/CISP-BMEI.2018.8633266.

42. KRUŽEJ, Martin. *Rozšíření Sparkle o grafickou tvorbu a vizualizaci dotazů: Extension of Sparkle to graphical building queries*. Západočeská univerzita v Plzni, 2019. Dostupné také z: http://hdl.handle.net/11025/37419. diplomová práce. Vedoucí práce Ing. VČELÁK Petr.

Seznam zkratek

`API` - Application Programming Interface
`CSV` - Comma Separated Values
`HTTP` - Hypertext Transfer Protocol
`HTTPS` - HTTP Secure
`IRI` - Internationalized Resource Identifier
`JSON` - JavaScript Object Notation
`LD` - Linked Data
`MVC` - Model View Controller
`ORM` - Object Relational Mapping
`OWL` - Web Ontology Language
`RDF` - Resource Description Framework
`RDFS` - RDF Schema
`REST API` - Representational State Transfer API
`SPARQL` - SPARQL Protocol and RDF Query Language
`SQL` - Structured Query Language
`URI` - Uniform Resource Identifier
`URL` - Uniform Resource Locator
`XML` - Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů