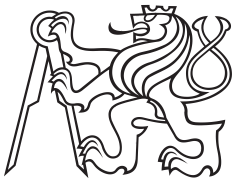
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Interaktivní dashboard pro RDF

Michal Švagr

Vedoucí: Ing. Petr Křemen, Ph.D.
Studijní program: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Švagr** Jméno: **Michal** Osobní číslo: **483856**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Interaktivní dashboard pro RDF

Název bakalářské práce anglicky:

Interactive dashboard over RDF

Pokyny pro vypracování:

Zatímco pro databázové technologie existuje celá řada nástrojů pro tvorbu interaktivních dashboardů, pro propojená data vyjádřená v jazyce RDF takové nástroje prakticky nejsou. Cílem práce je navrhnout a implementovat systém pro tvorbu interaktivních dashboardů nad RDF.
1) Seznamte se se standardy propojených dat - RDF, SPARQL - a stávajícími nástroji pro tvorbu dashboardů (např. Kibana)
2) Navrhněte architekturu systému pro tvorbu interaktivních dashboardů nad RDF daty dostupnými v SPARQL endpointech
3) Navrhněte webovou aplikaci umožňující konfigurovat interaktivní dashboardy nad vybranými RDF zdroji
4) Implementujte navrženou architekturu a webovou aplikaci
5) Vytvořte interaktivní dashboardy pro dvě vybrané datové sady, a vyhodnoťte efektivitu celé pipeline pomocí automatizovaných testů.

Seznam doporučené literatury:

[1] Colazzo, Dario & Goasdoué, François & Manolescu, Ioana & Roatiş, Alexandra. (2014). RDF Analytics: Lenses over Semantic Graphs. WWW 2014 - Proceedings of the 23rd International Conference on World Wide Web. 467-478. 10.1145/2566486.2567982.
[2] RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 2014, https://www.w3.org/TR/rdf11-concepts/
[3] SPARQL 1.1. Query Language. W3C Recommendation 2013, https://www.w3.org/TR/sparql11-query/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Křemen, Ph.D., skupina znalostních softwarových systémů FEL

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

Datum zadání bakalářské práce: **12.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

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

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych poděkovat panu Ing. Petru Křemenovi, Ph. D. za vedení, odbornou pomoc, věcné připomínky a poskytnutí příležitosti toto téma zpracovávat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 21. května 2021

Abstrakt

Tato bakalářská práce se zabývá problematikou analýzy a vizualizace RDF dat, což je jazyk pro popis dat a ontologií na sémantickém webu, a to ve formě grafů. Práce obsahuje stručné seznámení s použitými technologiemi, popisuje sestavení a návrh architektury řešící danou problematiku a prvotní testování efektivity této architektury. Výsledkem práce je implementace architektury systému pro indexaci RDF dat, správu těchto indexů, a jejich využití pro tvorbu interaktivních dashboardů s využitím systémů Kibana a Elasticsearch.

Klíčová slova: RDF(S), Dashboard, SPARQL, Kibana, Elasticsearch

Vedoucí: Ing. Petr Křemen, Ph.D.

Abstract

This bachelor thesis seeks to solve the problem of analysis and visualisation of RDF data, a language for describing data and ontologies on the Semantic Web in graphs. The work contains a brief introduction to the technologies used, describes the construction and design of the architecture and initial testing of the effectiveness of this architecture. The result of the work is an implementation of the architecture for indexing RDF data, management of these indexes, and their use for the creation of interactive dashboards using Kibana and Elasticsearch systems.

Keywords: RDF(S), Dashboard, SPARQL, Kibana, Elasticsearch

Title translation: Interactive dashboard over RDF

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

V dnešním světě jsme obklopeni daty, ale význam dat je často skrytý. K porozumění datům je zapotřebí analýza a nejlépe vizualizace ve srozumitelné formě. Dále můžeme data propojovat mezi sebou a popisovat pomocí již existujících pojmů. Před tento problém jsem byl postaven s méně známým, ale rychle se rozšiřujícím formátem dat Resource Description Framework (RDF) [8]. Tento druh dat je velice mocný hlavně v oblasti propojených dat, které můžeme vytvořit například z webových stránkách, propojení vědeckých prací nebo jiných publikací.

V současné době existuje velké množství RDF datasetů. Příkladem může být snad všem známá stránka `www.wikipedia.org`. Z jejích infoboxů[1] je automaticky extrahován strojově čitelný obsah ve formátu RDF nesoucí název DBPedia [23]. DBPedia poskytuje velký potenciál v analýze miliónů článků v mnoha jazycích, ale chybí jednoduchý způsob, jak data analyzovat. Pro analýzu je nyní nutná znalost dotazovacího jazyka SPARQL [38], který je běžnému uživateli nesrozumitelný, čímž se velice omezuje okruh dostupnosti těchto informací. Usnadněním v přístupnosti k informacím je vizualizace. V případě analýzy RDF dat by byl vhodný interaktivní dashboard schopný ukázat například propojenost jednotlivých zdrojů na `www.wikipedia.org`. Nebo vizualizovat statistiky všech pojmů vyskytujících se napříč zákony ČR a propojení jednotlivých zákonů ze vznikající eSbírky[2].

[1] Infobox je tabulka významných informací o textu.

[2] Informace o projektu `https://www.mvcr.cz/clanek/esbirka-a-elegislativa.aspx`

Příkladem takové analýzy muže být shromáždění všech výzkumných článků, epidemiologických statistik a klinických testů o COVID-19. Všechny tyto informace a jejich propojení jsou v podobě RDF dat zaznamenána, stačí si je pouze vypsat, uspořádat a zobrazit. Analýza a vizualizace těchto dat nám následně může ukázat shody ve studiích, změny v chování a projevech jednotlivých mutací, jak mutace rozeznat nebo účinnosti léků na jednotlivé mutace.

Již teď jsou na internetu volně dostupné nástroje pro vizualizaci RDF dat. Příkladem může být webová stránka SPARQL rozhraní `https://query.wikidata.org/`, poskytující možnost vizualizovat výsledek dotazu v sloupcovém nebo jiném grafu. Problémem těchto řešení je potřeba znalosti technologií RDF a SPARQL, složité použití a malá schopnost přizpůsobení vizualizace. Nástrojům také chybí možnost vizualizace ve více grafových formách najednou a interakce mezi jednotlivými grafy například pomocí filtrů.

Mým úkolem tedy bylo projít současně dostupné možnosti pro analýzu RDF datasetů a nástrojů na interaktivní dashboardy. Nalézt způsob propojení nástrojů obou technologií. A následně navrhnout architekturu pro implementaci takového řešení, které by usnadnilo analýzu RDF dat těm, kteří tyto technologie již ovládají, a zpřístupnilo získané informace většímu okruhu lidí.

Kapitola 2

Seznámení s technologií RDF

Tato kapitola se věnuje popisu technologie RDF [8] a z ní odvozeného ontologického jazyka RDF Schema [3]. Kapitola je zakončena komentovaným příkladem názorné ukázky použití technologie.

2.1 RDF

RDF neboli Resource Description Framework je obecný jazyk pro reprezentaci dat v podobě grafu, popisující zdrojový dokument způsobem pochopitelným jak pro člověka, tak strojově, specifikovaný organizací World Wide Web Consortium (W3C). RDF data jsou v grafovém formátu s orientovanými hranami, takže se dají velice jednoduše vizualizovat orientovanými grafy. Pomocí RDF lze popsat jakýkoliv zdroj, který lze identifikovat pomocí IRI [33] (International Resource Identifier). Nejznámějším druhem IRI jsou URL [37] (Uniform Resource Locators), které se používají pro adresaci webových stránek.

2.1.1 Trojice

V RDF jsou data popsána pomocí trojic ve tvaru subjekt - predikát - objekt. Subjekt a objekt představují dva zdroje nejčastěji identifikované pomocí IRI.

Predikát v trojici vyjadřuje vztah mezi subjektem a objektem ve směru subjekt → objekt, také pomocí IRI. Pouze na pozici objektu mohou být kromě IRI i literály neboli přímo zapsané hodnoty (např. řetězec). Specifikace také umožňuje prázdné uzly, jinými slovy subjekt nebo objekt může být v trojici neurčitý. Příklad takových trojic je názorně ukázán v části 2.3 Ukázka RDF na konci kapitoly.

2.2 RDF(S)

Z RDF je odvozen jednoduchý ontologický jazyk RDF Schema (RDFS nebo RDF-S) vyvinutý pod dohledem W3C. Je důležité zmínit, že v rozlišování RDF a RDF Schema je sice určena jasná hranice, ale není stanovena zrovna šťastně občas až nelogicky. Například, jak je vidět z prvních dvou řádků tabulky 2.1, konstrukt `rdfs:Class`[1] je definován v RDF Schema, ale konstrukt typu `rdf:Property`[2] je definován v RDF. Proto se často v textech setkáme se zkratkou RDF(S), která zastupuje jak pojem RDF, tak RDF Schema.

2.2.1 Syntax

Jak už je definováno v RDF, pro zápis se používají trojice subjekt - predikát – objekt. RDF(S) k tomu přidává konkrétní syntax [17], jako jsou například pravidla odvozování (doplňující do grafu data, která vyplývají z původního zápisu) nebo sedm hlavních předdefinovaných konstruktů, které pokládají základní kámen popisu dat:

Konstrukt	Syntaktická forma	Popis
Class (třída)	C rdf:type rdfs:Class	C (zdroj) je RDFS třída
Property (třída)	P rdf:type rdf:Property	P (zdroj) je RDF vlastnost
type (vlastnost)	I rdf:type C	I (zdroj) je instance C (třída)
subClassOf (vlastnost)	C1 rdfs:subClassOf C2	C1 (třída) je podřazená třída C2 (třída)
subPropertyOf (vlastnost)	P1 rdfs:subPropertyOf P2	P1 (vlastnost) je podřazená vlastnost P2 (vlastnost)
domain (vlastnost)	P rdfs:domain C	doména P (vlastnost) je C (třída)
range (vlastnost)	P rdfs:range C	rozsah P (vlastnost) je C (třída)

Tabulka 2.1: Základní konstrukty RDF(S)

[1] Zkrácený zápis IRI <http://www.w3.org/2000/01/rdf-schema#Class>
[2] Zkrácený zápis IRI <http://www.w3.org/1999/02/22-rdf-syntax-ns#Property>

2.2.2 Literály

Mohou se vyskytovat v trojici pouze na pozici objektu a vyjadřují konkrétní hodnoty. Literál je pro správnou interpretaci spjat s datovým typem, jako je například řetězec, číslo, desetinné číslo nebo datum. Řetězcové literály mohou ještě být spjaty s jazykovou značkou, například "Spiderman"@en. V našem příkladu (2.3 Ukázka RDF) si můžeme všimnout literálů na řádku 11 (s typem datum) a 15 (typu řetězec).

```
4  PREFIX schema: <http://schema.org/>
5  PREFIX dcterms: <http://purl.org/dc/terms/>
11       schema:birthDate  "1991-09-04"^^xsd:date ;
15       dcterms:title "Mona Lisa" ;
```

2.2.3 Zápis RDF(S) grafů

Pro zápis existuje velké množství formátů serializace. Většina z nich se, kromě serializovaného zápisu, snaží i o zkrácení zápisu. Například Turtle umožňuje zastoupení úvodní částí IRI prefixy. Mezi nejpoužívanější serializace se řadí tyto:

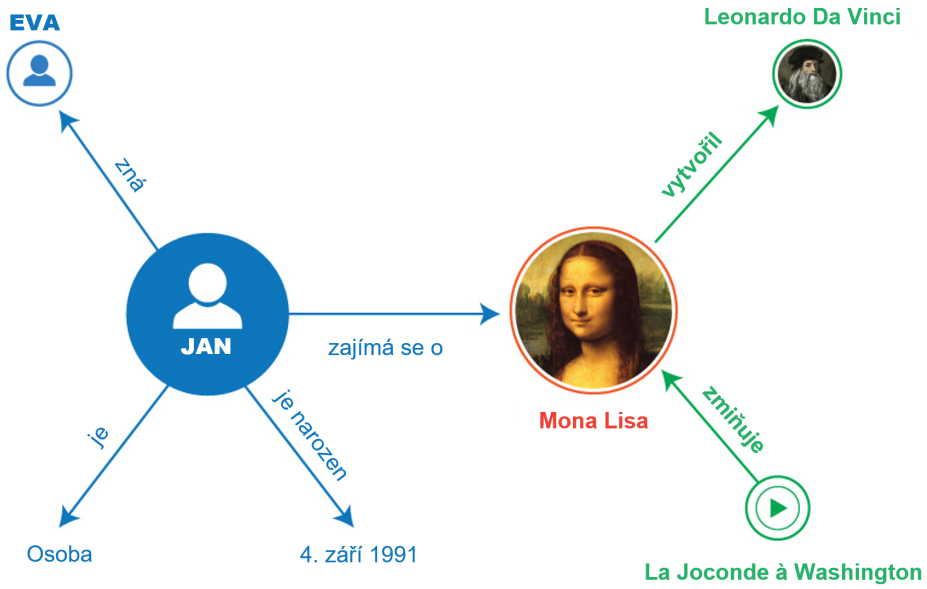
- Rodina Turtle jazyka RDF (N-Triples [31], Turtle [29], TriG [5] a N-Quads [4])
- JSON-LD [21] (RDF syntaxe založena na JSON notaci)
- RDFa [2] (pro vyjádření RDF uvnitř HTML dokumentů)
- RDF/XML [14] (XML syntax pro RDF)

2.3 Ukázka RDF

```
1  BASE   <http://ukazka.net/>
2  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
3  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4  PREFIX schema: <http://schema.org/>
5  PREFIX dcterms: <http://purl.org/dc/terms/>
6  PREFIX wd: <http://www.wikidata.org/entity/>
7  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
8
9  <jan#me>
10         a foaf:Person ;
11         foaf:knows <eva#me> ;
12         schema:birthDate  "1991-09-04"^^xsd:date ;
13         foaf:topic_interest wd:Q12418 .
14
15 wd:Q12418
16         dcterms:title "Mona Lisa" ;
17         dcterms:creator <http://dbpedia.org/resource/Leonardo_da_Vinci> .
18
19 <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
20         dcterms:subject wd:Q12418 .
21         rdfs:label "la Joconde à Washington" .
```

Příklad je psán ve formátu Turtle, kde první řádek nastavuje pomocí BASE základní adresu pro dokument. Dalších 5 řádků nastavují prefixy pro zkrácení zápisu v dokumentu. Řádek 8 začíná blok popisu pojmu `http://ukazka.net/jan#me`. Následující odsazené řádky nastavují třídu na `Person`, vztah `"knows"` s `http://ukazka.net/eva#me`, vlastnost `"birthDate"` literál `"1991-09-04"` typu datum a vztah `"topic_interest"` a IRI obrazu "Mona Lisa". Řádky 14 až 16 reprezentují vlastnosti obrazu. Nejdříve vlastnost `"title"` a následně vztah `"creator"` s IRI Leaonarda Da Vinciho. Poslední tři řádky tvoří dvě trojice, a to IRI videa (kde se vyskytuje obraz "Mona Lisa") ve vztahu `"subject"` s IRI obrazu "Mona Lisa"a vlastnost `"label"` videa.

Celý graf nám tedy říká, že existuje osoba Jan narozená 4. září 1991, která zná Evu a zajímá se o obraz Mona Lisa. Tvůrce tohoto obrazu je Leonardo da Vinci. Zmínka o obraze Mona Lisa je také ve videu s názvem La Joconde à Washington. Graf těchto dat by tedy mohl vypadat takto:



Obrázek 2.1: Graf příkladu RDF(S)

Kapitola 3

Dotazovací jazyk SPARQL

SPARQL [38] neboli Simple Protocol and RDF Query Language je sémantický dotazovací jazyk na data ve formátu RDF vytvořen skupinou RDF DAWG (Data Access Working Group) zaštítěnou organizací W3C. Tento standard je uznáván jako jedna z klíčových technologií sémantického webu. SPARQL tedy uživatelům umožňuje formulovat dotazy na koncové úložiště (endpoint), podobně jako by se jednalo o databázi, a dále výsledek dotazu zpracovávat (např. agregací). V této kapitole jsou ukázány pouze základy jazyka potřebné pro práci se vzniklou architekturou.

3.1 Typy dotazů

Dotazy ve SPARQL jsou definovány ve čtyřech formách, následovány blokem WHERE omezujícím dotaz:

- SELECT

  Používá se k výpisu nezměněných dat z koncového bodu SPARQL (SPARQL endpoint), výsledky jsou vráceny ve formátu tabulky.

- CONSTRUCT

  Používá se k výpisu informací z koncového bodu SPARQL s převedeným výsledkem do formátu RDF.

- ASK

  Používá se k výpisu odpovědi ano/ne z dotazu na koncový bod SPARQL.

- DESCRIBE

  Používá se k popisu užitečných informací o webovém zdroji. Konkrétní výběr užitečných informací je na rozhodnutí SPARQL. Častou strategií je vracet trojice, kde subjektem je popisovaný webový zdroj. Tento formát má jako jediný volitelný blok WHERE.

3.2 Syntax

Tato sekce pokrývá základní syntax v jazyce SPARQL pro dotazování RDF dat. Úplný syntax naleznete v materiálech W3C o SPARQL dotazech [16] v bodě 19.

3.2.1 Proměnné a PREFIXy

Proměnné jsou v SPARQL dotazích tvořeny pomocí prefixu ? nebo $ před jménem proměnné. Pro zkrácení zápisu se používají prefixové konstanty ve formátu *PREFIX jmeno_prefixu: <IRI_prefixu>*. Například následující dotaz vrátí tabulku o sloupcích "x" a "name", kde sloupec "x" budou identifikátory IRI a v druhý sloupec "name" bude obsahovat jméno (hodnotu vlastnosti http://www.w3.org/2001/vcard-rdf/3.0#FN):

```
1 PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
2
3 SELECT ?x ?name
4 WHERE {?x  vcard:FN  ?name}
```

x	name
<http://example.cz/janhole01/>	"Jan Hole"
<http://example.cz/matejpotok/>	"Matěj Potok"
<http://example.cz/evakopecna/>	"Eva Kopečná"

Tabulka 3.1: SPARQL příklad - proměnné

3.2.2 Filtry

Podobně jako v SQL existuje WHERE, v SPARQL existuje parametr FILTER selektující řádky výstupní tabulky pomocí boolovské podmínky. FILTER podporuje mimo jiné i filtrování za pomocí regulárních výrazů, porovnávání hodnot nebo ověření typu. Následuje příklad jednoduchého filtru vybírající osoby starší 18 let. Dotaz opět vypíše dva sloupce, v prvním jméno a v druhém věk:

```
1  PREFIX info:         <http://somewhere/peopleInfo#>
2  PREFIX vcard:       <http://www.w3.org/2001/vcard-rdf/3.0#>
3
4  SELECT ?name ?age
5  WHERE
6  {
7      ?person vcard:FN  ?name .
8      ?person info:age ?age . FILTER ( ?age > 18 )
9  }
```

name	age
"Jan Hole"	20
"Eva Kopečná"	42

Tabulka 3.2: SPARQL příklad - Filtry

3.2.3 OPTIONAL

OPTIONAL označuje možnost zobrazit v tabulce i ty řádky, které nemají všechny požadované sloupce, tedy analogie LEFT OUTER JOIN v SQL. Například kdybychom v předchozím příkladu přidali na 8. řádek OPTIONAL, tak budou vypsány všechny osoby, ale jen u osob starších 18 let bude napsán věk:

```
1  PREFIX info:         <http://somewhere/peopleInfo#>
2  PREFIX vcard:       <http://www.w3.org/2001/vcard-rdf/3.0#>
3
4  SELECT ?name ?age
5  WHERE
6  {
7      ?person vcard:FN  ?name .
8      OPTIONAL { ?person info:age ?age . FILTER ( ?age > 18 ) }
9  }
```

name	age
"Jan Hole"	20
"Matěj Potok"	
"Eva Kopečná"	42

Tabulka 3.3: SPARQL příklad - OPTIONAL

3.2.4 UNION

UNION umožňuje spojovat více slovníků do jedné výsledné odpovědi dotazu. Například následující dotaz spojí a vypíše všechna jména uložená ve slovnících `foaf` a `vCard` do jednoho sloupce výsledné tabulky:

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
3
4 SELECT ?name
5 WHERE
6 {
7    { [] foaf:name ?name } UNION { [] vCard:FN ?name }
8 }
```

name
"Jan Hole"
"Matěj Potok"
"Eva Kopečná"
"Tomáš Straka"
"Anna Dvořáková"
"Barbora Novotná"

Tabulka 3.4: SPARQL příklad - UNION

Kapitola 4

Rešerše současného stavu

V této kapitole jsou popsány vlastnosti vyhledaných technologii, které by bylo možné použít pro řešení problematiky práce. Významné nalezené technologie jsou zde shrnuty a porovnány s ostatními.

4.1 RDF nativní technologie

Při hledání technologii jsem nejdříve vyhledával takové, které podporovaly RDF nativně. Jako jediné slibné řešení se jevila SANSA [30] (Scalable Semantic Analytics Stack) používající Apache Flink [36] a Apache Spark [24].

4.1.1 SANSA

SANSA je vytvořena pro rychlé zpracování velkých semantických datasetů. Bohužel při bližším zkoumání technologie a pokusech o lokální zprovoznění se ukázalo, že dokumentace technologie je velice omezená a již samotnou instalaci provázelo spoustu potíží. S přihlédnutím k tomu, že RDF je pro mne úplně nová technologie a její zpracování také, je pro mě taková dokumentace nedostatečná. Další nevýhodou je omezené množství dostupných vizualizačních nástrojů.

4.2 Technologie bez nativní podpory RDF

Změnil jsem tedy způsob hledání na dashboardy s podporou pluginů, což by mi umožnilo doimplementovat propojení analytických a RDF funkcionalit. Z nalezených technologií mě nejvíce zaujaly nástroje Tableau [35], Chart.js [6], Apache SuperSet [1] a Kibana [11].

4.2.1 Tableau

Jedná se o dashboardovací nástroj s analytickými prvky. Modul Tableau Prep umožnuje kombinování více zdrojů dohromady, přetváření a filtrování dat. Zároveň umožnuje automatizovat přípravy dat v grafickém rozhraní. Bohužel je produkt placený a to metodou měsíčního předplatného za každého uživatele. Tato skutečnost znamená, že Tableau komunita se skládá převážně z firem a ty své know-how a řešení potíží volně nešíří. Dokumentace a podpora je stavěna na míru zákazníkovi, což je určitě dobré pro velké firmy, ale ne pro mě jakožto jedince snažící se integrovat toto řešení s novou technologií.

4.2.2 Chart.js

Jak už z názvu vyplývá Chart.js je JavaScriptová knihovna pro vytváření grafů dostupná pod open source licencí MIT. Za touto knihovnou stojí velice rozšířená komunita a dokumentace je detailní s příklady. Řešení tímto způsobem by zahrnovalo vytvoření i interaktivity a propojení na dashboardu a to již některé nástroje umí.

4.2.3 Apache SuperSet

Opět se jedná o webový nástroj pro tvorbu analytických dashboardů s velikou škálou vizualizačních grafů stejně jako Tableau. Na rozdíl od Tableau je Apache SuperSet open source software s detailní dokumentací, rozsáhlým API[1] a velkou komunitou. Jediné co Apache SuperSet chybí, je komunita zajímající se o technologii RDF.

[1] API neboli Application Programming Interface je rozhraní umožnující propojení s jinými softwary

4.2.4 Kibana

Nástroj Kibana od společnosti Elastic [9] je v mnoha ohledech stejný jako Apache SuperSet. Skvělá detailní dokumentace, rozsáhlé API rozhraní a open source frontend aplikace s rozmanitými vizualizačními grafy. V čem ale Kibana předčívá Apache SuperSet je RDF komunita. Po nedlouhém hledání jsem narazil na projekt s názvem EEA ElasticSearch RDF Indexer [34]. Tento projekt, již od počátečních verzí Kibany, umožňuje indexovat RDF data, která se následně dají vizualizovat a analyzovat v nástroji Kibana.

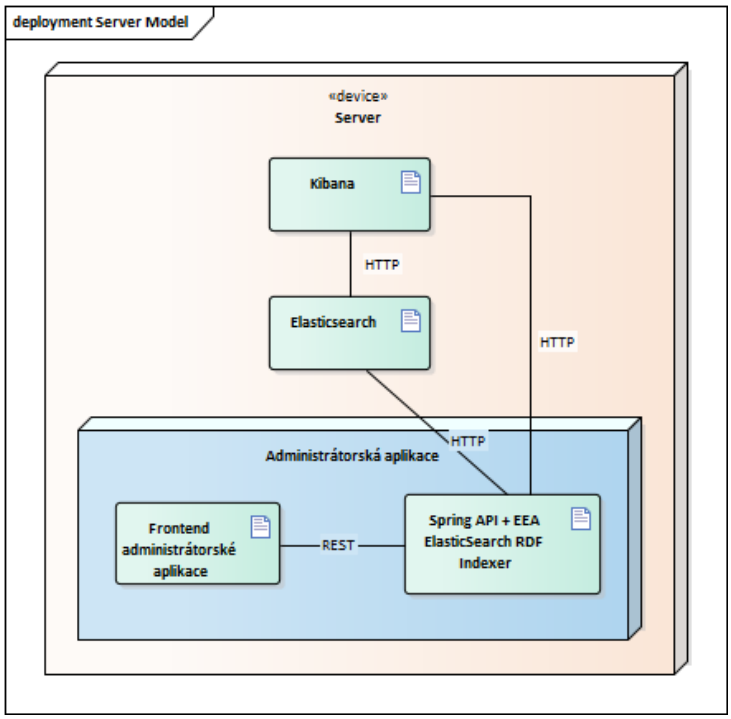
4.3 Závěr rešerše

Nástroj SANSA dle svých specifikací vypadá jako velmi mocný analytický nástroj pro práci s RDF daty, nebýt chudé dokumentace jedná se pravděpodobně o nejlepší nástroj pro tuto práci díky jeho nativní integraci s RDF daty. Tableau se jeví jako vhodný nástroj pro velké firmy. Personalizovaná podpora slibuje vysvětlení jak postupovat krok za krokem na příkladu podobném zákazníkovu přání. Chart.js s vytvořením vlastního dashboardu a analytické části je určitě nejvíce přizpůsobitelná konkrétnímu použití, ale vytvoření bude stát spoustu času. Nástroje Apache Superset a Kibana jsou si opravdu velice podobné, ale Kibana byla pro mne jasnou volbou kvůli již zmíněnému projektu EEA ElasticSearch RDF Indexer. A díky velkému rozšíření produktů od společnosti Elastic bude integrace pro stávající analytiky jednodušší.

Kapitola 5

Architektura

Navrhovaná architektura se skládá z dashboardovacího nástroje Kibana [11], vyhledávacího nástroje Elasticsearch [10], indexovací aplikace EEA ElasticSearch RDF Indexer a administrátorské aplikace pro ovládání indexování.



Obrázek 5.1: Model nasazení architektury na serveru

5.1 Elasticsearch

Elasticsearch je open source vyhledávací modul od společnosti Elastic schopný zpracovávat textová, číselná, geoprostorová, strukturovaná a nestrukturovaná data. Poskytuje jednoduché REST API umožňující lehké propojení s jinými aplikacemi. Elasticsearch je centrálním modulem Elastic Stack, neboli sady open source nástrojů k přijímání, obohacování, ukládání, analýze a vizualizaci dat.

5.2 Kibana

Další modul z již zmiňovaného Elastic Stack umožňující analýzu a vizualizaci dat. Stejně jako u Elasticsearch se jedná o open source aplikaci zpracovávající indexovaná data v Elasticsearch. Vizualizace je zde uskutečněna pomocí dashboardů s grafy, canvasy, mapami, filtry pro čištění dat a grafovými nástroji pro vizualizaci propojení mezi daty. Díky nativnímu propojení těchto dvou technologií Kibana zároveň přidává grafické rozhraní pro správu indexů v Elasticsearch.

5.3 EEA ElasticSearch RDF Indexer

Jak už název napovídá, jedná se o software od agentury Evropské unie European Environment Agency (EEA) pro vytváření indexů RDF dat komunikující s Elasticsearch pomocí jeho API. Původně vytvořený jako plugin přímo do Elasticsearch a následně předělaný jako samostatná javovská aplikace pro Elasticsearch ve verzi 6.1 a vyšší. Hlavní funkce aplikace:

- indexování dat ze souborů a SPARQL dotazů typu SELECT, CONSTRUCT a DESCRIBE,
- vlastní pojmenování vytvořených indexů v Elasticsearch,
- normalizaci vlastností indexování,
- přidávání nepotřebných vlastností na černou listinu,
- přidávání vyžadovaných vlastností na bílou listinu,
- synchronizaci s koncovým bodem,
- plánování sběru dat

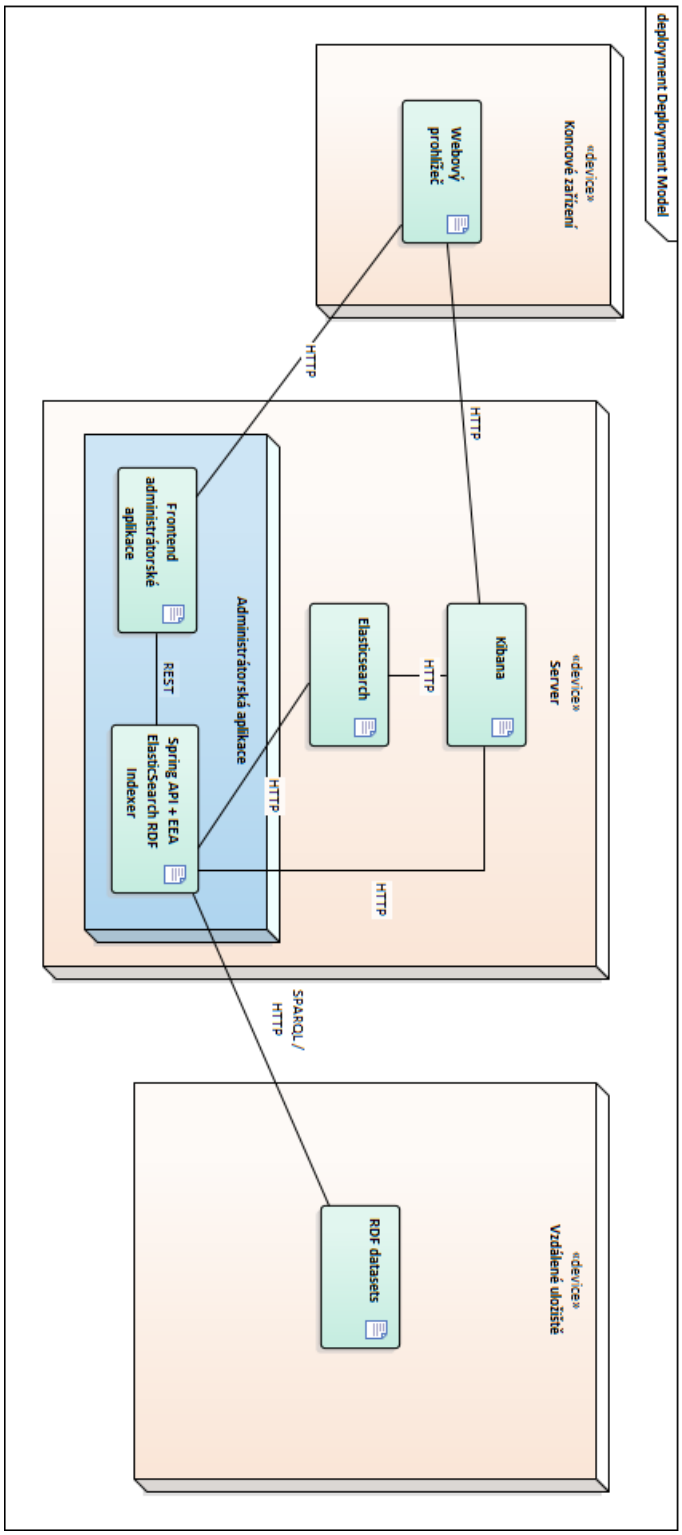
5.4 Administrátorská aplikace

Administrátorská aplikace se skládá ze dvou částí. První částí je úprava a rozšíření aplikace EEA ElasticSearch RDF Indexer o API server, neboli o komunikační rozhraní umožňující ovládání upraveného Indexeru. A druhou částí je klientská aplikace usnadňující vytváření, úpravu a mazání konfigurací indexování a monitorování běhu Indexeru. Komunikace mezi klienstskou aplikací a komunikačním rozhraním serveru probíhá za pomocí REST dotazů přes HTTP. Detailní popis administrátorské aplikace naleznete v kapitole 6 Administrátorská aplikace.

5.5 Výhody, nevýhody a komunikace v architektuře

Jak je vidět z obrázku 5.2 na další straně, centrálním uzlem architektury je Administrátorská aplikace, která ukládá konfigurace indexací, spouští indexace, zasílá naindexovaná data do Elasticsearch, načítá informace o propojení indexů a dashboardů z Kibany a poskytuje webové uživatelské rozhraní. Elasticsearch následně poskytuje naindexovaná data Kibaně, která poskytuje webové dashboardovací nástroje. Všechna komunikace mezi těmito technologiemi probíhá pomocí HTTP požadavků.

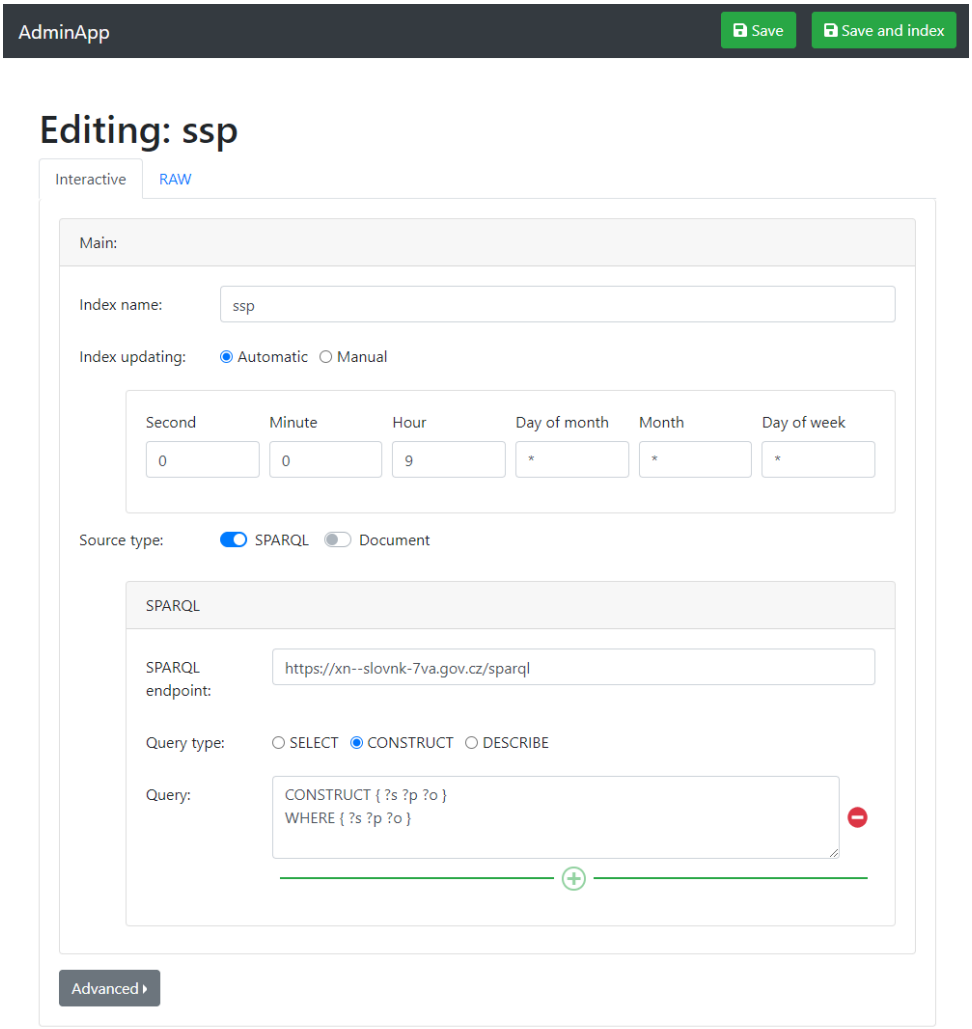
To poskytuje, společně s detailní dokumentací, rozsáhlou komunitou a množstvím nástrojů Elastic Stacku, velký potenciál k rozšiřitelnosti. Kde naopak dokumentace lehce zaostává je aplikace EEA ElasticSearch RDF Indexer. Tuto skutečnost však napravuje přístup ke zdrojovému kódu aplikace a tedy možnosti úpravy a analýzy kódu. Další výhodou je možnost indexování SPARQL dotazů, které je aplikace EEA ElasticSearch RDF Indexer schopna sama zasílat a zpracovat, stejně tak indexování souboru ve vzdálených uložištích.



Obrázek 5.2: Model komunikace v architektuře

5.6 Příklad komunikace

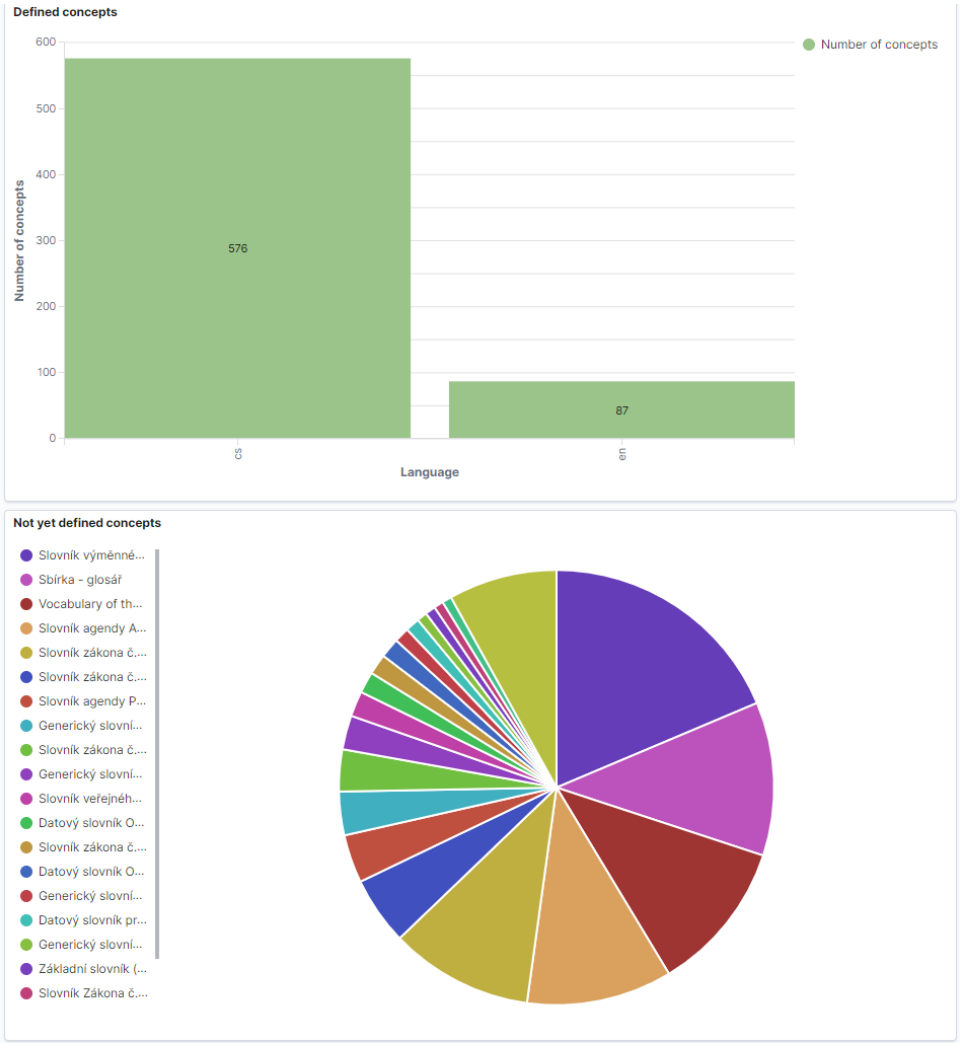
Kdybychom například chtěli indexovat nová data, začali bychom v administrátorské aplikaci přípravou konfigurace Indexeru, která obsahuje zdroje dat, název indexu k vytvoření, typ dat, plán aktualizací a případně pokročilé informace k indexaci. Například takto:



Obrázek 5.3: Nastavení konfigurace v administrátorské aplikaci

Tato konfigurace se následně převede do JSON formátu a bude uložena do serverové části administrátorské aplikace. Při manuálním zapnutí indexace, nebo když nadejde naplánovaný čas, administrátorská aplikace spustí Indexer. Při indexování jsou data ze vzdáleného úložiště postupně stažena do Indexeru a následně přetvořena do indexované podoby. Takto indexovaná data jsou

zaslána do dočasného indexu v Elasticsearch a po úspěšném nahrání zaměněna za nastavený index. Index se automaticky zobrazí v grafickém rozhraní Kibany a po vytvoření šablony indexu už lze data vizualizovat a analyzovat.



Obrázek 5.4: Vizualizace na indexovaných datech

Kapitola 6

Administrátorská aplikace

Tato kapitola se věnuje tvorbě administrátorské aplikace. Popisuje návrh aplikace od architektury po low-fidelity prototyp, hlavní funkce, využité technologie a nasazení do kontejnerového prostředí docker.

6.1 Návrh aplikace

Nejprve jsem navrhl architekturu, sepsal požadavky a vytvořil prototyp Administrátorské aplikace.

6.1.1 Architektura

K vytvoření administrátorské aplikace jsem využil architektury klient-server. Serverovou částí je Spring bootová [32] nadstavba nad aplikací EEA ElasticSearch RDF Indexer. Jako klientskou část jsem vytvořil novou React [18] aplikaci. Pro komunikaci klienta se serverem používám RESTové [12] rozhraní se serializací do JSON [7] formátu.

6.1.2 Funkční požadavky

Tato sekce obsahuje požadavky na softwarovou stránku administrátorské aplikace.

- FP1 – Systém bude umožňovat zobrazení všech spravovaných konfigurací indexování s časem poslední aktualizace a délkou trvání.
- FP2 – Systém bude umožňovat zobrazení indexem ovlivněných dashboardů a odkaz na tyto dashboardy.
- FP3 – Systém bude umožňovat procházení alespoň posledních tří záznamů o aktualizaci indexu.
- FP4 – Systém bude umožňovat zobrazení průběhu indexování v reálném čase.
- FP5 – Systém bude umožňovat vytvoření nové konfigurace indexování pomocí uživatelsky přívětivých formulářových polí včetně pokročilých vstupů, kterými jsou například normalizace vlastností, filtrovací mapa, popis URI, atd.
- FP6 – Systém bude umožňovat uživateli zadat konfiguraci ve stávající podobě JSON formátu EEA ElasticSearch RDF Indexeru.
- FP7 – Systém bude umožňovat úpravu existujících konfigurací indexování.
- FP8 – Systém bude umožňovat smazání existujících konfigurací indexování.
- FP9 – Systém bude umožňovat spuštění indexace dle nastavené konfigurace.
- FP10 – Systém bude umožňovat zastavení běžící indexace.
- FP11 – Systém bude umožňovat naplánované spuštění indexace dle nastavené konfigurace.

6.1.3 Prototyp

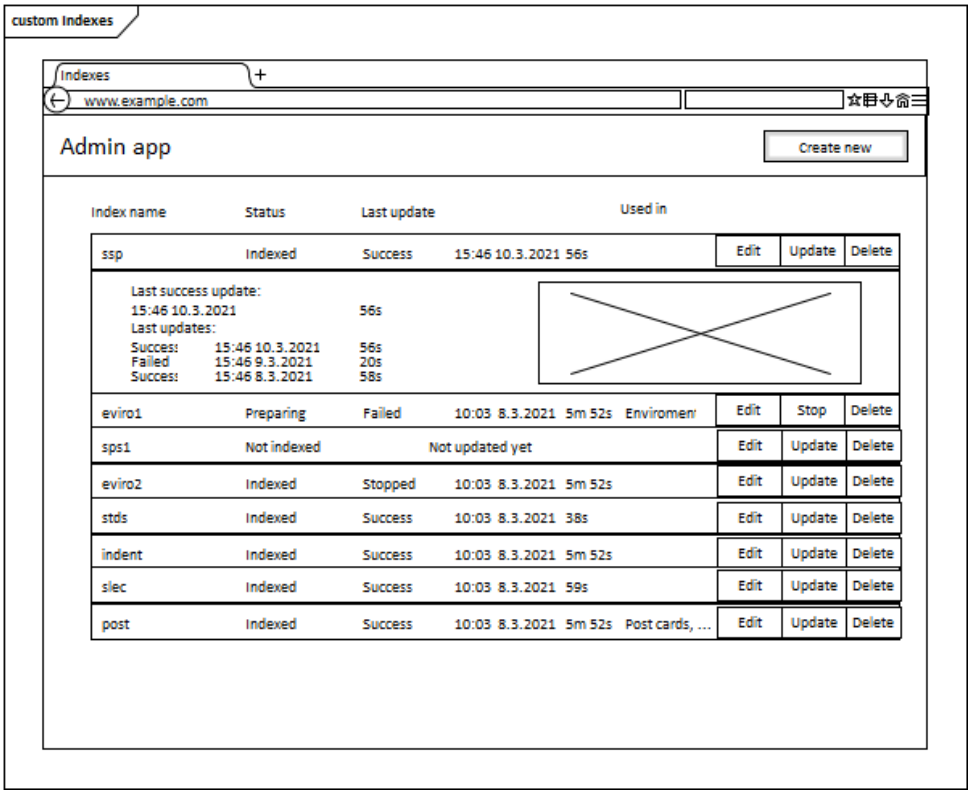
Frontendový prototyp jsem vytvořil v low-fidelity (nízké přesnosti) vzhledem k rozsáhlosti a přímočarosti aplikace. První, úvodní obrazovka, ukazuje přehledné informace o všech konfiguracích indexací. Druhá obrazovka „Úprava konfigurace“ umožňuje úpravu případně vytvoření nové konfigurace indexování.

Úvodní obrazovka

Jakožto úvodní obrazovku jsem navrhl tabulku všech konfigurací indexování. Každý řádek tabulky obsahuje: název výsledného indexu, status indexu v reálném čase, záznam o poslední aktualizaci indexu, seznam všech dashboardů kde se index používá a tři tlačítka.

- "Edit" pro úpravu konfigurace, které ukáže obrazovku Úprava konfigurace.
- "Update" na manuální spuštění aktualizace nebo zastavení běžící indexace.
- "Delete" pro smazání indexu ukáže uživateli potvrzující pop-up s informacemi co smazání ovlivní a zda chce uživatel smazat i naindexovaná data.

Každý řádek také obsahuje skryté dodatečné informace, kde nalezneme: poslední úspěšnou aktualizaci, seznam posledních aktualizací a graf ukazující jak se délky aktualizací od sebe navzájem liší.



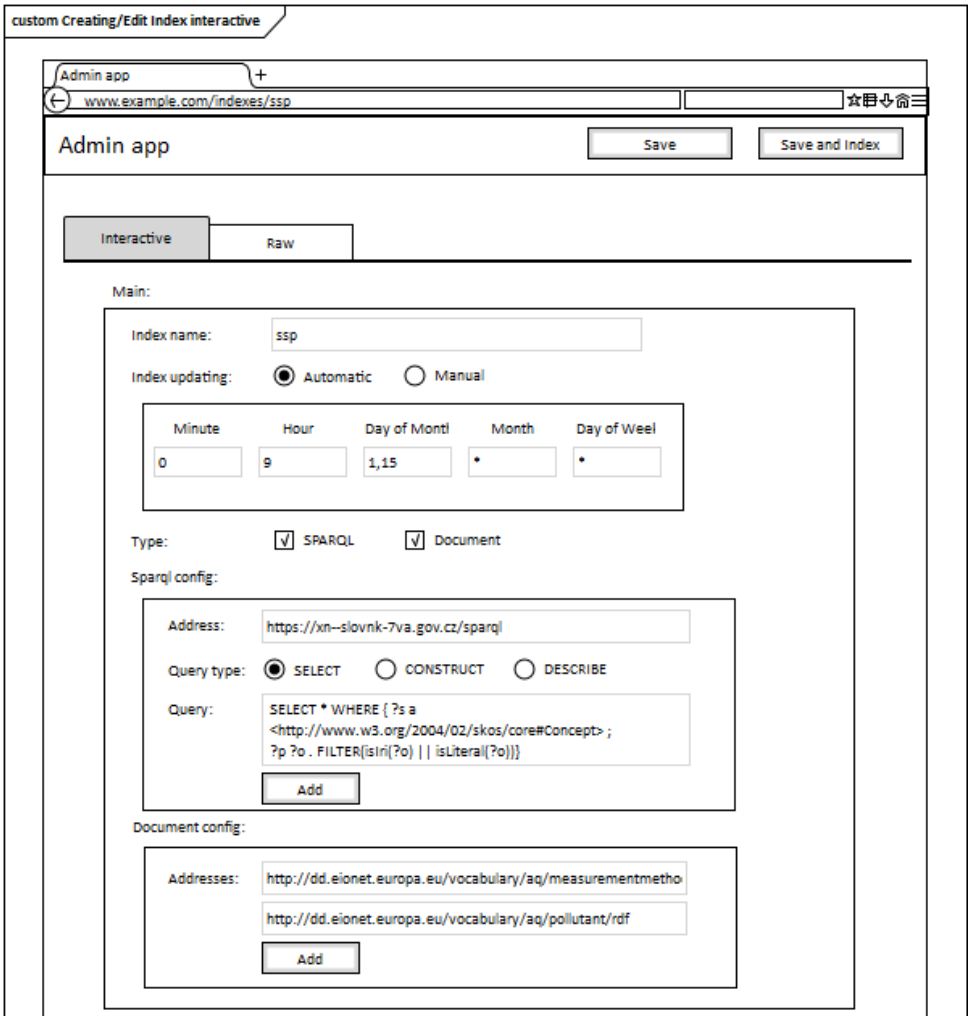
Obrázek 6.1: Wireframe úvodní obrazovky

Úprava konfigurace

Tato obrazovka umožňuje uživateli upravit nebo vytvořit konfiguraci indexování pomocí dvou módů úpravy, mezi kterými může uživatel plynule přecházet. Interaktivní mód je přehlednější s nápovědami a jak název napovídá interaktivní. Mód přímého vstupu je určen pro pokročilé uživatele nebo vložení již existujících konfigurací EEA ElasticSearch RDF Indexer.

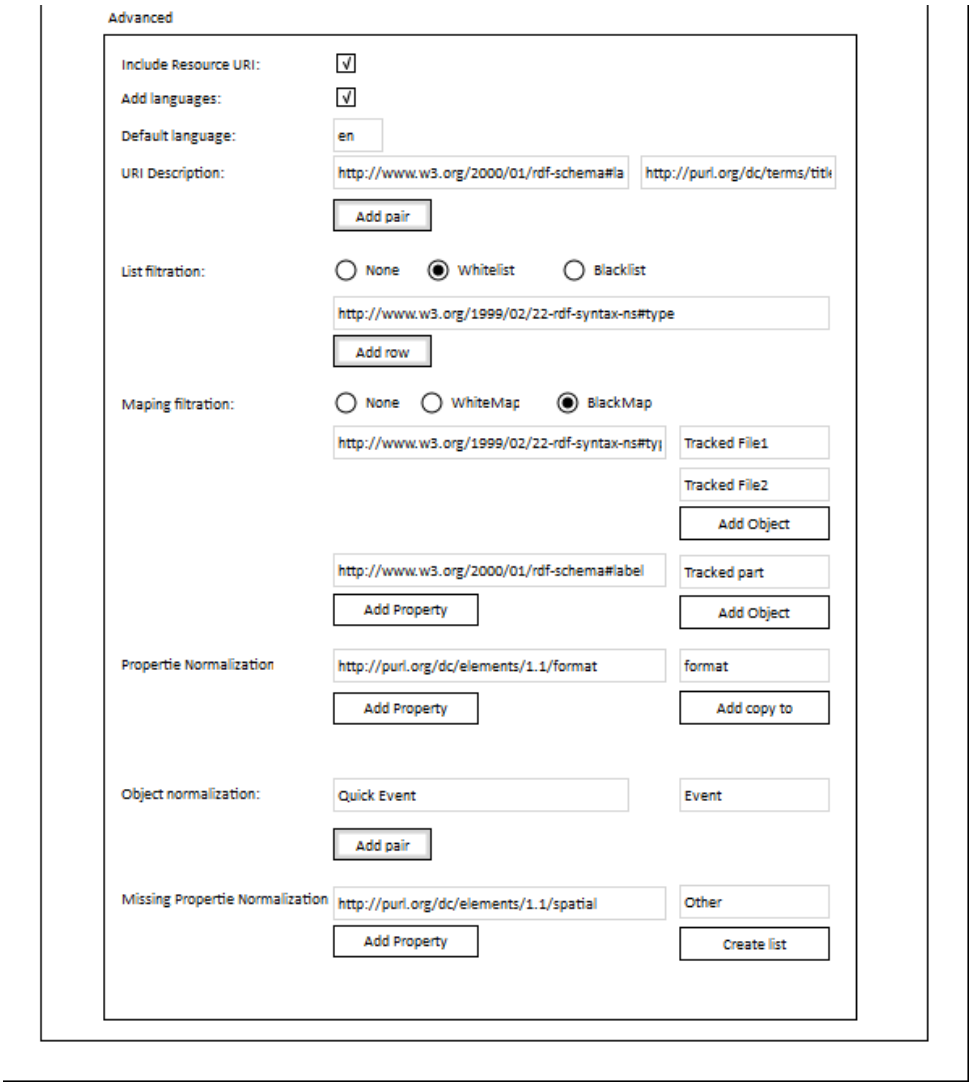
- Interaktivní konfigurace

  V interaktivní části jsou jednotlivá pole ke konfiguraci indexování zobrazena pomocí formulářových prvků. Začíná potřebnými poli: název výsledného indexu, frekvence aktualizací ve formátu CRON [22] a definování zdrojů dat.



Obrázek 6.2: Úprava konfigurace - interaktivní

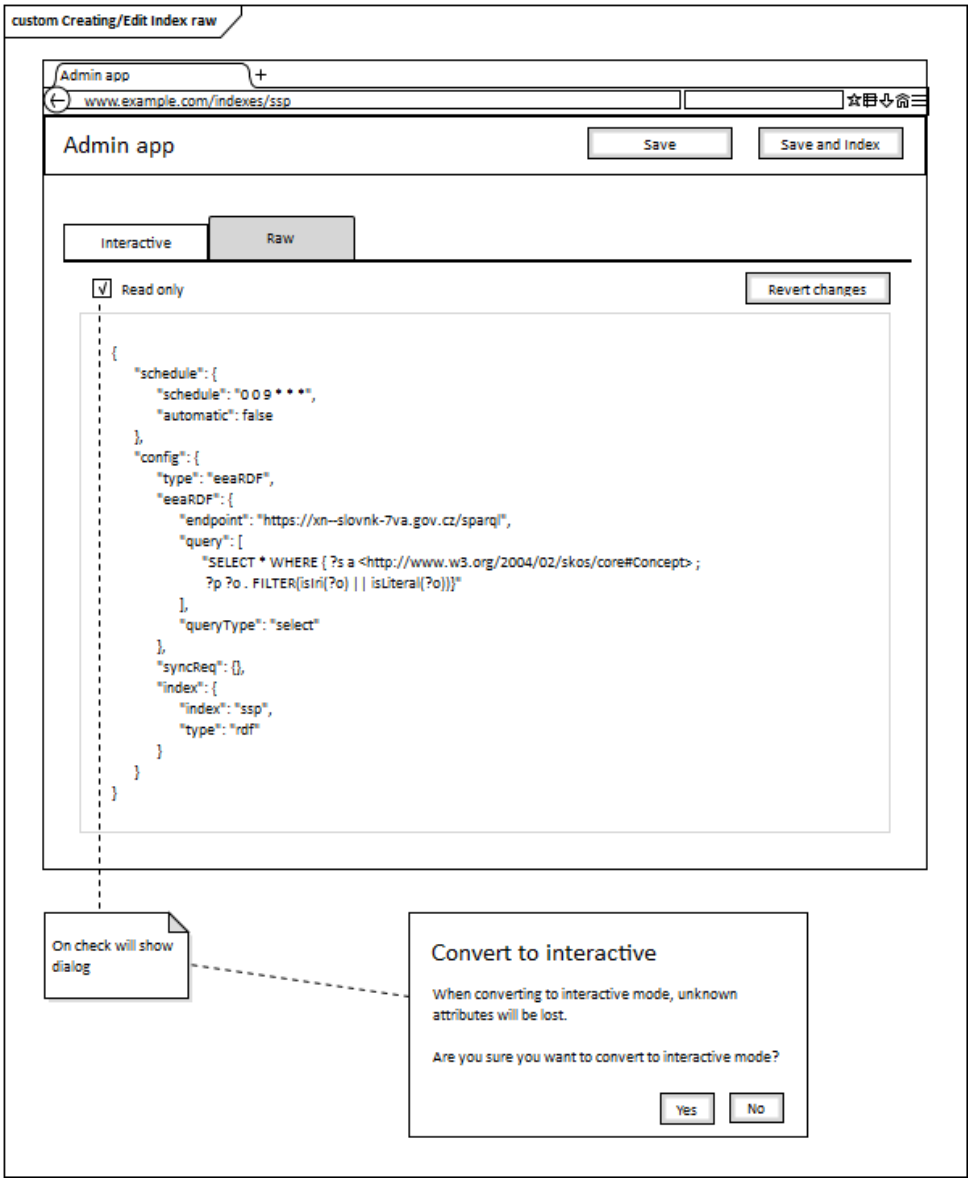
Navazuje rozbalovací část zobrazující pokročilé možnosti konfigurace. Zde jsou nastavovány pravidla pro úpravu indexu v době indexování, jako je: zahrnutí zdrojových URI do indexu, přidání i jiných jazyků než výchozího, výchozí jazyk, nastavení vlastnosti popis URI, filtrování zdrojů, filtrování mapovaných spojení, přejmenování vlastností v indexu, pravidla pro ucelení termínů a nastavení výchozích hodnot chybějících vlastností pojmů.



Obrázek 6.3: Úprava konfigurace - rozšířené možnosti interaktivní

- Konfigurace pomocí přímého vstupu

Druhým způsobem je konfigurace pomocí napsání JSON struktury minimálně s parametry: plán aktualizace, zapnutí automatické aktualizace, adresu vzdáleného bodu, pole dotazů, typ dotazu, název indexu a popis typu dat. Kvůli omezenosti interaktivního módu je zde pop-up upozornění uživatele, na ztrátu neznámých parametrů, při návratu do interaktivního módu. Dále tento způsob konfigurace umožňuje uživateli návrat k původní hodnotě.



Obrázek 6.4: Úprava konfigurace - přímý vstup

6.2 Technická analýza

Tato část popisuje hlavní funkce, úpravu EEA ElasticSearch RDF Indexeru a použité technologie v administrátorské aplikaci.

6.2.1 Frontend

Pro tvorbu uživatelského rozhraní aplikace jsem vybral React, kvůli jeho rozšířenosti a jednoduchosti použití. Komunikaci se serverem obstarává knihovna Axios [25] usnadňující odesílání dotazů a vyhodnocování odpovědí. Grafickou stránku frontendu usnadňují knihovny: React Bootstrap [19] svými předpřipravenými responzivními CSS [20] styly, Recharts [15] tvořící jednoduché grafy a React Font Awesome 4 [13] poskytující SVG [39] ikony.

6.2.2 Backend

Jak už je zmíněno v 6.1.1 Architektura, pro serverovou část jsem si vybral Spring boot a to hned z několika důvodů. EEA ElasticSearch RDF Indexer je napsán v Javě [27] a tedy mohu pouze tuto aplikaci rozšířit o Spring boot, což usnadňuje práci a vylučuje Elasticsearch jakožto prostředníka v předávání informací o konfiguracích a indexování. Zároveň mám se Spring boot již zkušenosti.

Při vytváření Spring Boot nadstavby bylo zapotřebí také upravit EEA ElasticSearch RDF Indexeru, aby splňoval všechny požadavky.

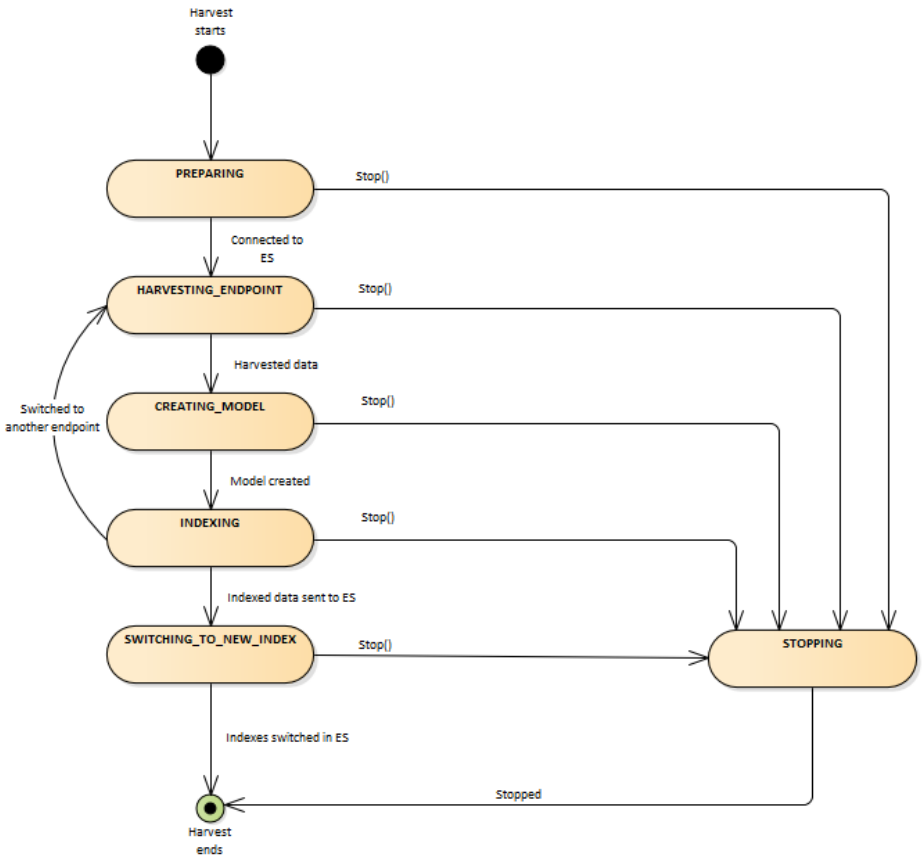
Rollback indexování

EEA ElasticSearch RDF Indexer naindexovaná data začne zasílat okamžitě do Elasticsearch kde jsou záznamy přidány nebo upraveny pokud již existují. Zastavení indexování by nám tedy nechalo změny v indexu. Samotná aktualizace indexu sice upraví stávající záznamy a přidá nové, ale neodebere záznamy reprezentující data, která z koncového bodu zmizela.

Naimplementoval jsem tedy funkcionalitu, která indexuje data do takzvaného dočasného indexu a až na konci úspěšného indexování nahradí současný index za ten dočasný. Tudíž při zastavení nebo nepovedeném indexování se pouze provede rollback (smaže se dočasný index) a aktualizace je zcela nezávislá na předchozích záznamech.

Monitorování indexování

Indexer také nezaznamenává, v jaké části procesu indexování se nachází a neukládá výsledek indexování. Doimplementoval jsem tedy změny stavů (viz diagram níže) v Indexeru a vytvořil rozhraní, které umožňuje zjištění stavu indexování a jeho zastavení. Přidal jsem také ukládání výsledku a doby indexování.



Obrázek 6.5: Stavy indexování

Optimalizace

Backend je přizpůsoben pro využití z více zařízení najednou díky cacheování náročně získatelných dat (jako jsou například informace o propojení indexů a dashboardů) z Elasticsearch a Kibana. Výkon serveru je také rozložen na více vláken procesoru a to tak, že každý příchozí dotaz a indexování je zpracován na novém vláknu. Backend také přepočítává většinu dat do vhodného formátu pro rychlejší odezvu a uživatelovu práci na frontendu.

Chyby v Indexeru

Při rozšiřování aplikace EEA ElasticSearch RDF Indexer o serverovou část se ukázalo, že vydaná verze Indexeru není zcela hotová a bez chyb. V kódu byla spousta poznámek TODO (ještě dopracovat), ale drtivá většina měla příznak DONE (hotovo). A poznámky, které příznak neměly, byly doporučení na optimalizaci. Neočekával jsem tedy při prvním pokusu o indexování v Indexeru potíže. Indexer téměř okamžitě ukázal ukazoval úspěšné naindexování, ale v Elasticsearch se nový index neobjevil. Ukázalo se, že v aplikaci jsou některé výjimky, které přeruší celé indexování, pouze odchyceny, nevypsány a následně indexace skončí. Tyto chyby se postupně objevovaly po celou dobu práce. Další chyby se ukázaly v implementacích rozšířených možností konfigurace Indexeru, kde některé fungovaly pouze částečně nebo v nich byla jednoduchá logická chyba. Chyby jsem však postupně opravoval a vznikl velice stabilní Indexer.

6.3 Nasazení aplikace

Administrátorskou aplikaci nasazuji do docker kontejneru s prostředím Java 8 společně s kontejnery Elasticsearch a Kibany. Díky balíčkování do Java jar je však možné aplikaci nasadit v jakémkoliv prostředí s nainstalovaným Java Runtime Environment 8 [28].

Jako první krok je zapotřebí sestavit produkční build frontendové aplikace pomocí balíčku npm:

```
1 npm install
2 npm run build
```

Build následně vložíme k serverové části do *frontend/build* a serverovou aplikaci zkompilujeme:

```
1 mvn clean install
```

Tím nám vznikne balíček serveru obsahující i frontend. Následně stáhneme obrazy Elasticsearch a Kibany:

```
1 docker pull docker.elastic.co/elasticsearch/elasticsearch:7.10.0
2 docker pull docker.elastic.co/kibana/kibana:7.10.0
```

K nasazení už jen stačí upravit soubor *docker-compose.yml* , kde jsou uloženy informace o portech jednotlivých docker kontejnerů a vytvořit kontejnery:

```
1 docker compose up
```

Kapitola 7

Testování

V této kapitole jsou popsány metody, prostředí, výsledky a závěr testování efektivity vytvořené architektury. Testovány jsou dva různé datasety pomocí odlišných způsobů sběru dat z koncových bodů.

7.1 Testovací prostředí

Všechny testy proběhly v kontejnerovém prostředí docker nastaveného dle 6.3 Nasazení aplikace společně s Elasticsearch a Kibanou. Specifikace systému[1]:

Verze docker	20.10.5, build 55c4c88 (WSL 2)
Operační systém	Windows 10 Pro
Procesor	Intel core i5-8250U
RAM	8 GB (DDR4)
Druh disku	SATA SSD
Internetové připojení	
Odezva	22ms
Rychlost stahování	15mb/s
Rychlost nahrávání	4mb/s

Tabulka 7.1: Specifikace testovacího prostředí

[1] Agregovaná odezva k SPARQL endopintu `https://query.wikidata.org/`

7.2 Metoda testování

Jsou testovány dvě metody sběru dat. První pomocí dotazu SPARQL SELECT a druhá pomocí SPARQL CONSTRUCT, která používá stejný způsob indexování jako při sběru ze souborů. Měřen je počet řádků výsledku dotazu, počet naindexovaných záznamů v Elasticsearch, doba trvání testu od uložení konfigurace, k úplnému nahrání indexu do Elasticsearch a doba sběru dat a indexace.

V obou metodách je nejdříve zkontrolováno připojení k Elasticsearch a Kibana. Následně je vytvořena konfigurace indexování a odeslána na Spring server. Server zahájí indexování a test začne simulovat pravidelné dotazy frontendu. Po skončení indexace je zkontrolován výsledek indexace, zaznamenány výsledky a index smazán z Elasticsearch.

7.3 Testované datasety

Pro metodu SPARQL SELECT jsou sbírána data o všech zaznamenaných nemocích a jejich příznacích z RDF úložiště Wikidata [26] (`https://www.wikidata.org`), které sdružuje data vytvářená člověkem nebo roboty z jejich sesterských projektů jako jsou Wikipedia, Wikivoyage, Wiktionary, Wikisource a další.

SPARQL CONSTRUCT metoda indexuje data z Sémantického slovníku pojmů (`http://xn--slovnk-7va.gov.cz`). Slovník popisuje pojmy české legislativy a významové vztahy mezi nimi. Slouží k vyhledávání, dokumentaci a metadatovému popisu datových sad a datových rozhraní veřejné správy. Tento dataset je vhodný pro testování zejména kvůli jeho velikosti a rychlému rozšiřování.

7.4 Výsledky testování

V této sekci jsou vypsány výsledky všech testů efektivity. Každá tabulka s výsledky obsahuje počet řádků výsledku SPARQL dotazu, výsledný počet záznamů v Elasticsearch po indexaci, celkovou dobu trvání, průměrný čas na

zpracování jednoho řádku a průměrný čas na vygenerování jednoho záznamu. Časy jsou zaznamenány jak na celý test, tak na samotný sběr dat s následnou indexací.

SPARQL SELECT

Testováno nad daty o nemocích a jejich příznacích. Každý dotaz je omezen pomocí LIMIT na daný počet řádků výsledku dotazu.

- SPARQL SELECT – 1 000 řádků

Počet řádků	1 000
Výsledný počet záznamů	883
Celý test	
Celkový čas	2s 268ms
Čas za řádek	2ms (2,268ms)
Čas za záznam	2ms (2,569ms)
Sběr a indexování	
Celkový čas	2s 222ms
Čas za řádek	2ms (2,222ms)
Čas za záznam	2ms (2,516ms)

Tabulka 7.2: Výsledek efektivity 1 000 řádkového SELECT dotazu

- SPARQL SELECT – 5 000 řádků

Počet řádků	5 000
Výsledný počet záznamů	1 671
Celý test	
Celkový čas	8s 154ms
Čas za řádek	1ms (1,631ms)
Čas za záznam	4ms (4,880ms)
Sběr a indexování	
Celkový čas	7s 774ms
Čas za řádek	1ms (1,555ms)
Čas za záznam	4ms (4,652ms)

Tabulka 7.3: Výsledek efektivity 5 000 řádkového SELECT dotazu

- SPARQL SELECT – 10 000 řádků

Počet řádků	10 000
Výsledný počet záznamů	1 672
Celý test	
Celkový čas	8s 357ms
Čas za řádek	0ms (0,836ms)
Čas za záznam	4ms (4,998ms)
Sběr a indexování	
Celkový čas	8s 295ms
Čas za řádek	0ms (0,830ms)
Čas za záznam	4ms (4,961ms)

Tabulka 7.4: Výsledek efektivity 10 000 řádkového SELECT dotazu

SPARQL CONSTRUCT

Testováno nad daty o pojmech z české legislativy a významovými vztahy mezi nimi. Každý dotaz je omezen pomocí LIMIT na daný počet řádků výsledku dotazu.

- SPARQL CONSTRUCT – 1 000 řádků

Počet řádků	1 000
Výsledný počet záznamů	896
Celý test	
Celkový čas	41s 752ms
Čas za řádek	41ms (41,752ms)
Čas za záznam	46ms (46,598ms)
Sběr a indexování	
Celkový čas	41s 686ms
Čas za řádek	41ms (41,686ms)
Čas za záznam	46ms (46,525ms)

Tabulka 7.5: Výsledek efektivity 1 000 řádkového CONSTRUCT dotazu

- SPARQL CONSTRUCT – 5 000 řádků

Počet řádků	5 000
Výsledný počet záznamů	4 517
Celý test	
Celkový čas	51s 498ms
Čas za řádek	10ms (10,300ms)
Čas za záznam	11ms (11,401ms)
Sběr a indexování	
Celkový čas	51s 418ms
Čas za řádek	10ms (10,284ms)
Čas za záznam	11ms (11,383ms)

Tabulka 7.6: Výsledek efektivity 5 000 řádkového CONSTRUCT dotazu

- SPARQL CONSTRUCT – 10 000 řádků

Počet řádků	10 000
Výsledný počet záznamů	9 517
Celý test	
Celkový čas	1m 4s 634ms
Čas za řádek	6ms (6,463ms)
Čas za záznam	6ms (6,791ms)
Sběr a indexování	
Celkový čas	1m 4s 578ms
Čas za řádek	6ms (6,458ms)
Čas za záznam	6ms (6,786ms)

Tabulka 7.7: Výsledek efektivity 10 000 řádkového CONSTRUCT dotazu

- SPARQL CONSTRUCT – 20 000 řádků

Počet řádků	20 000
Výsledný počet záznamů	16 651
Celý test	
Celkový čas	6m 25s 959ms
Čas za řádek	19ms (19,298ms)
Čas za záznam	23ms (23,179ms)
Sběr a indexování	
Celkový čas	6m 25s 895ms
Čas za řádek	19ms (19,295ms)
Čas za záznam	23ms (23,175ms)

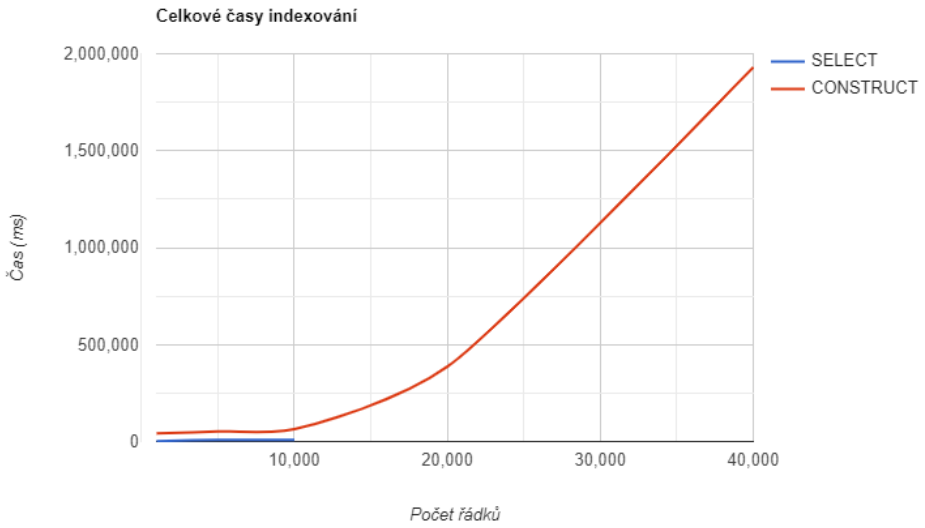
Tabulka 7.8: Výsledek efektivity 20 000 řádkového CONSTRUCT dotazu

- SPARQL CONSTRUCT – 40 000 řádků

Počet řádků	40 000
Výsledný počet záznamů	18 193
Celý test	
Celkový čas	32m 9s 983ms
Čas za řádek	48ms (48,250ms)
Čas za záznam	106ms (106,084ms)
Sběr a indexování	
Celkový čas	32m 9s 629ms
Čas za řádek	48ms (48,241ms)
Čas za záznam	106ms (106,064ms)

Tabulka 7.9: Výsledek efektivity 40 000 řádkového CONSTRUCT dotazu

7.5 Závěr testování efektivity



Obrázek 7.1: Graf celkových časů testování

Indexování pomocí dotazů SPARQL SELECT je velmi rychlé a bez problému vyhovuje běžnému použití. Avšak příprava SELECT dotazů je náročnější a vyžaduje větší znalost indexovaných dat. Oproti tomu jednoduše vytvořený SPARQL CONSTRUCT dotaz, je pomalejší v čase na zpracování jednoho řádku výsledku dotazu a to průměrně desetinásobně. Testování také naznačuje, že data menší deseti tisícům řádků téměř neovlivňují celkovou dobu zpracování a u dat větších dvaceti tisícům řádků doba zpracování roste úměrně přidaným řádkům. I když jsou SPARQL CONSTRUCT dotazy značně pomalejší, jsou stále vytvořeny správně a v přijatelném čase[2].

Dodatečné ověření v kódu také ukázalo na zdroj zdržení SPARQL CONSTRUCT metody oproti SPARQL SELECT metodě. Pro každý výsledný pojem u SPARQL CONSTRUCT jsou zasílány dotazy koncovému bodu o příslušný název (popisný literál) pojmu. Tento proces vždy čeká na odpověď koncového bodu až poté zašle další dotaz. Odezva koncového bodu je v řádu milisekund, ale dotaz je sekvenčně odesílán pro všechny pojmy a v případě nevrácení názvu je dotaz odeslán s jinými parametry až desetkrát.

[2]Značně kratší než frekvence aktualizací většiny RDF uložišť.

Kapitola 8

Přínosy vytvořené architektury

V současné době je možné analyzovat RDF data a to i za pomocí grafů. Příkladem může být webová stránka SPARQL rozhraní `https://query.wikidata.org/`, poskytující možnost vizualizovat výsledek dotazu v sloupcovém nebo jiném grafu. Je však zapotřebí mít rozšířenou znalost RDF a SPARQL. Prvním krokem by bylo prozkoumat strukturu RDF datasetu, vybrání požadované informace a vyhledání druhů spojení informace. Po prozkoumání struktury datasetu následuje napsání dotazu tak, aby výstup byl upraven pro vstupy grafu. Spuštění dotazu nám zobrazí chtěný graf. Pokud bychom chtěli graf z jiných dat, musíme celý proces opakovat a pracně vytvořený dotaz se nám nikam neuloží. SPARQL dotaz může pak vypadat například následovně[a]:

```
#defaultView:LineChart
SELECT ?age (COUNT(?person) AS ?count) WHERE {
{
SELECT ?person (SAMPLE(?age) AS ?age)  WHERE {
    ?person wdt:P509 wd:Q84263196 ; wdt:P31 wd:Q5
    OPTIONAL { ?person wdt:P570 ?d }
    ?person wdt:P569 ?dob ; wdt:P570 ?dod . BIND(YEAR(?dod)-YEAR(?dob) as ?age)

   SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
   }
 }
GROUP BY ?person
}
}
GROUP BY ?age
```

[a]Ukázku výsledného grafu naleznete na obrázku A.6

Vytvořená architektura tento proces zjednodušuje a rozšiřuje. Nejdříve v administrátorské aplikaci vytvoříme konfiguraci nového indexu pomocí webového formuláře (viz obrázek A.1 a A.2). Zde také zadáme adresu dokumentu nebo také SPARQL dotaz, ale můžeme zadat mnohem jednodušší a obecnější dotaz. Pro náš příklad použijeme dotaz, který nám naindexuje celý Sémantický slovník pojmů:

```
1 CONSTRUCT { ?s ?p ?o }
2 WHERE { ?s ?p ?o }
```

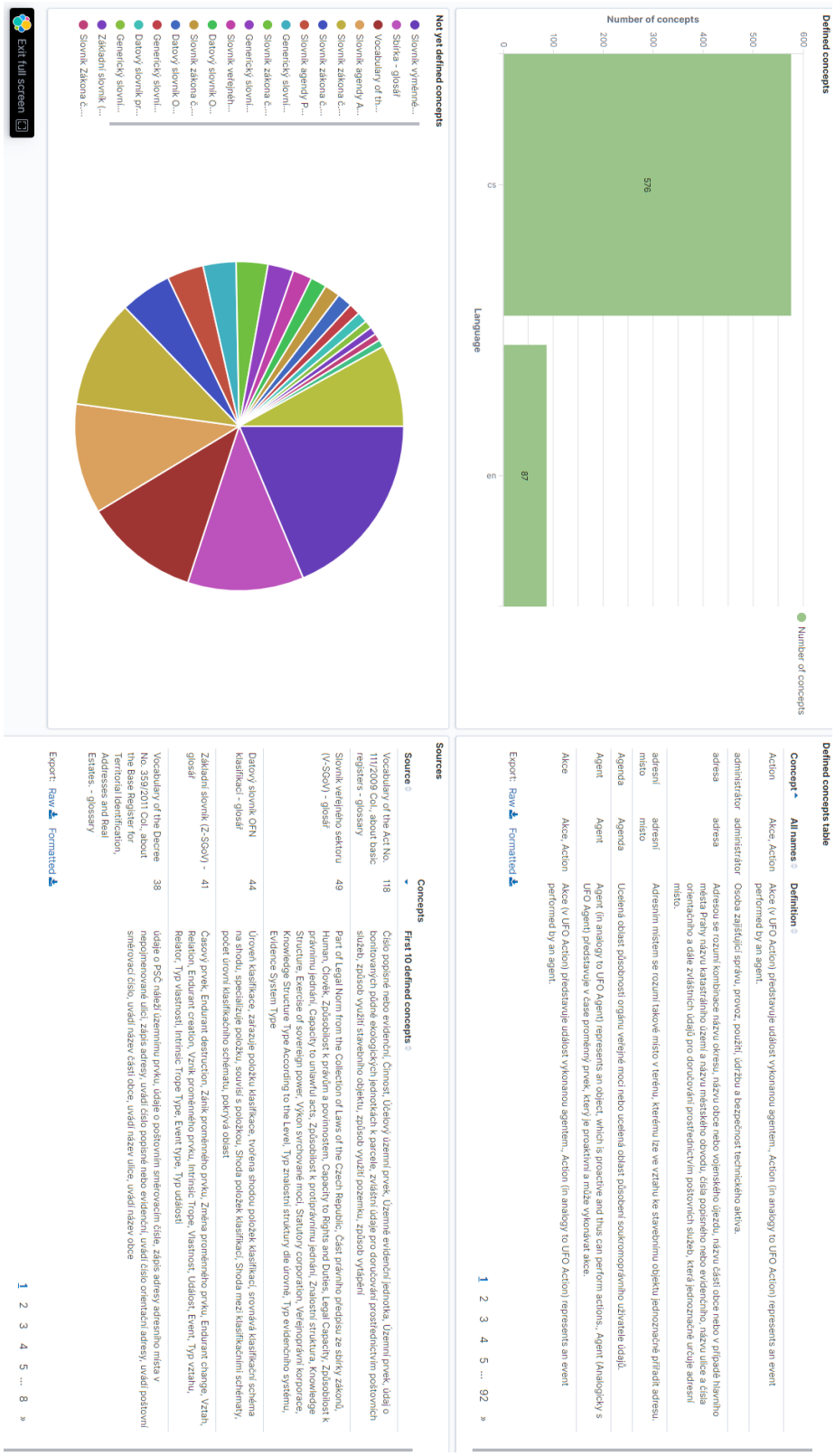
Po naindexování dat[1] se přesuneme do dashboardovacího nástroje Kibana. V menu *Stack Management* v sekci *Index patterns* si necháme automaticky vytvořit novou šablonu našeho indexu. Nyní můžeme prozkoumávat naindexovaná data v menu *Discover* (obrázek A.4) nebo rovnou začít vytvářet grafy v menu *Visualize*. Při vytváření si nejdříve vybereme druh grafu ze seznamu (sloupcový, spojnicový, plošný, koláčový, mapa, tabulka atd.), který lze rozšířit pomocí pluginů. Po vybrání nám Kibana ukáže interaktivní obrazovku pro vytváření grafu (obrázek A.5). Díky naindexování nám Kibana nabízí všechna dostupná pole a nemusíme je hledat ve struktuře datasetu. Vytvořený graf můžeme stejným způsobem i kdykoliv upravit.

Další výhodou architektury je možnost vytvoření více grafů a jejich vložení na dashboard. Tato funkcionalita nám umožňuje udržovat více grafů přehledně na jednom místě. A díky funkci automatických aktualizací indexovaných dat administrátorskou aplikací budou grafy samy aktualizovány.

8.1 Ukázka dashboardu SSP datasetu

Vytvořil jsem tedy pár grafů a tabulek z naindexovaných dat Sémantický slovník pojmů, který obsahuje české zákony a vyhlášky. Na sloupcovém grafu vlevo nahoře je vidět počet definovaných pojmů v jednotlivých jazycích. Koláčový graf pod ním ukazuje rozložení zatím nedefinovaných pojmů napříč jednotlivými dokumenty. Tabulka vpravo nahoře vypisuje všechny definované pojmy jejich názvy a následně definice. Poslední čtvrtinu dashboardu reprezentuje tabulka všech zdrojových dokumentů, počet definovaných pojmů z dokumentu a výpis definovaných pojmů.

[1] Ukázka naindexovaného indexu v aplikaci A.3

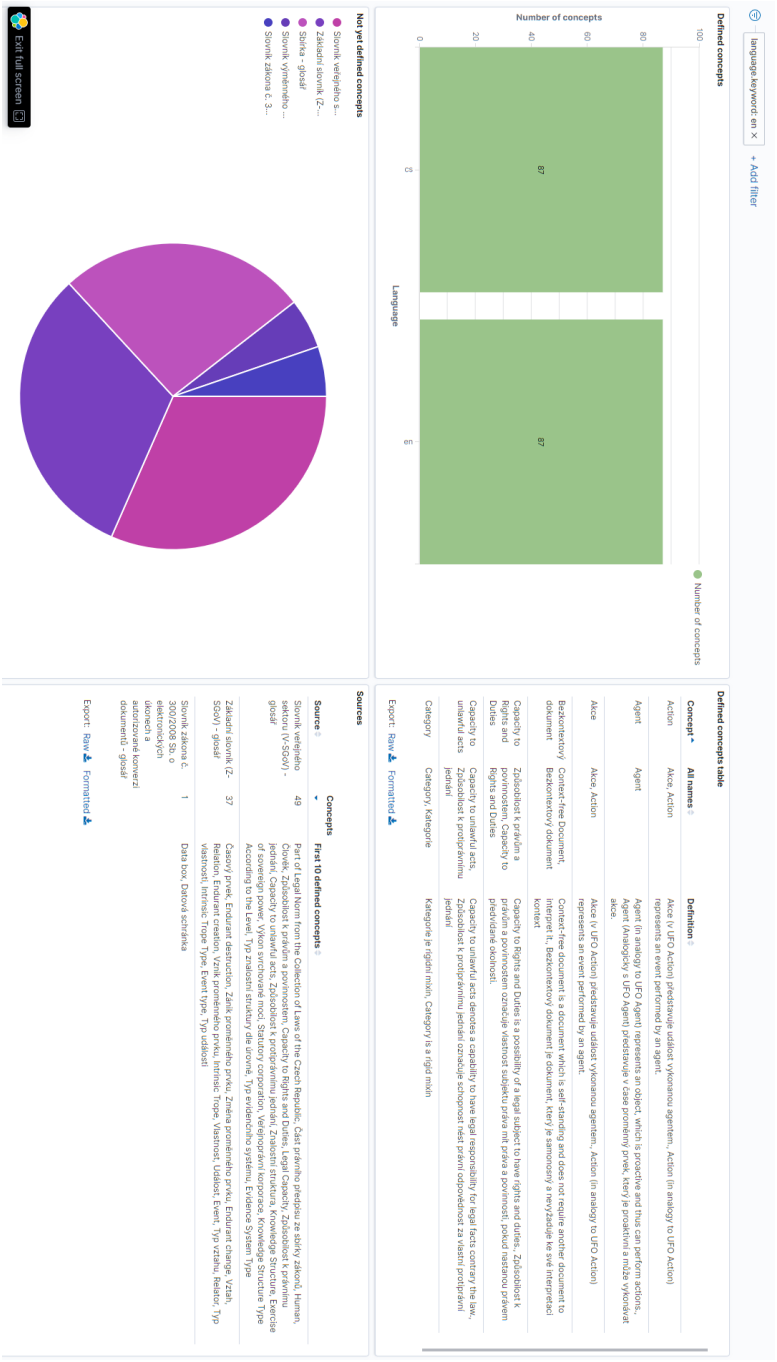


Obrázek 8.1: Dashboard datasetu SSP

Jednotlivé prvky na dashboardu jsou interaktivní. Například, když klikneme na jeden ze zdrojových dokumentů, na celý dashboard se aplikuje filtr pouze na tento dokument. Díky tomu hned vidíme, že všechny pojmy v Datovém slovníku OFN klasifikací jsou definovány pouze v češtině a existují pouze dva pojmy bez definice.

Obrázek 8.2: Dashboard datasetu SSP - vybraný zdrojový dokument

Nebo při kliknutí na sloupec anglického jazyka okamžitě vidíme, že všechny anglické pojmy jsou definovány i česky, nejvíce pojmů je ze Slovníku veřejného sektoru (V-SGoV). Také vidíme z koláčového grafu, že existují pojmy v anglickém jazyce, které zatím nejsou definované.



Obrázek 8.3: Dashboard datasetu SSP - filtrovaný anglický jazyk

8.2 Ukázka nad daty o nemocích

Pro druhou ukázku jsem vybral data o nemocích a jejich symptomech z již zmíněné stránky `https://query.wikidata.org/`, pomocí dotazu:

```
SELECT distinct (?disease as ?s) ?p (?symptom_label as ?o)
WHERE {
    ?disease wdt:P780 ?symptom .
    {
        ?symptom rdfs:label ?symptom_label .
        FILTER(lang(?symptom_label) = "en")
        SERVICE wikibase:label { bd:serviceParam wikibase:language "en".}
        Bind( wdt:P780 as ?p)
    } Union {
        ?disease rdfs:label ?symptom_label .
        FILTER (langMatches( lang(?symptom_label), "EN" ) )
        Bind(  rdfs:label as ?p)
    }
}
```

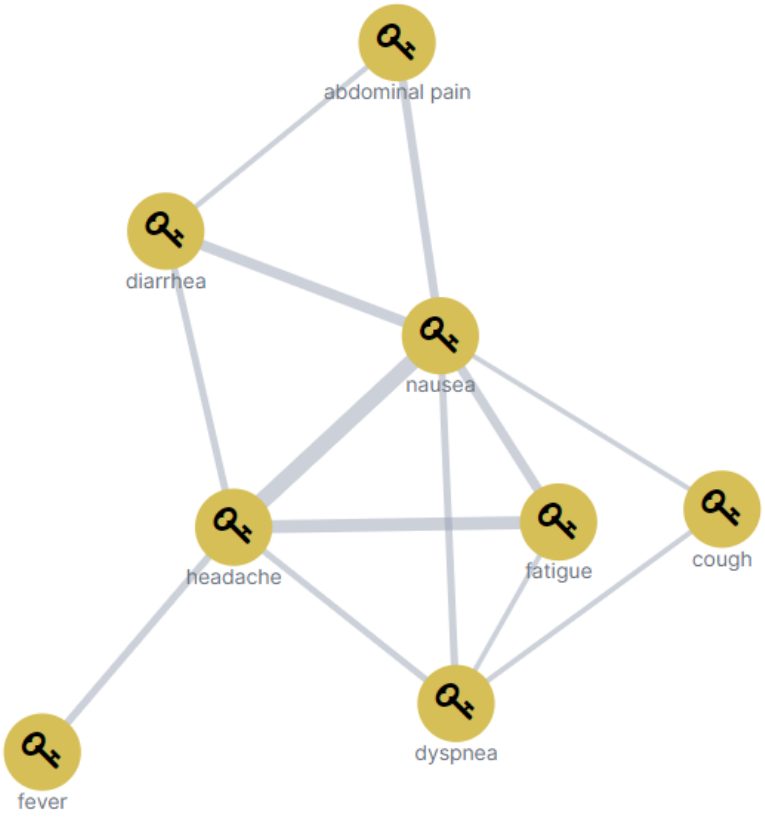
Vytvořený dashboard obsahuje tabulku nemocí, které mají společné příznaky s onemocněním COVID-19, v levém horním rohu. Vpravo od ní je sloupcový graf četnosti všech symptomů. Ve spodní části jsou vypsány všechny nemoci s jejich symptomy. Toto rozložení jsem vybral z důvodu velice intuitivního vyhledávání nemocí.

Obrázek 8.4: Dashboard nemocí

Kliknutím na jakýkoliv symptom se nám data omezí pouze na nemoci s tímto symptomem. To nám také ovlivní samotný graf symptomů. Například kliknutím na dva hlavní symptomy chřipky - kašel (cough) a horečka (fever), se nám nabízí jako nejčastěji spojovaný symptom bolest hlavy (headache), dušnost (dyspnea), průjem (diarrhea) a zimnice (chills). V tabulkách si můžeme všimnout mimo samotné chřipky (influenza) i nachlazení (common cold) a nemoci COVID-19.

Obrázek 8.5: Dashboard nemocí - s filtry

Pro tento druh dat je také dobrá vizualizace pomocí uzlového grafu. V našem příkladu se asi nejvíce hodí graf významných propojení jednotlivých symptomů. Na výsledném grafu můžeme vidět uzly jako jednotlivé symptomy a tloušťku hran jako sílu propojení s druhým symptomem. Také si můžeme všimnout, že dva nejčetnější symptomy, jak je vidět ze sloupcového grafu na ukázce *Dashboard nemocí*, podráždění oči (eye iritation) a podráždění kůže (irritant dermatitis) nejsou v uzlovém grafu významných propojení. To naznačuje skutečnosti, že přes jejich četnost nejsou silně vázány na jiný. Naopak zde můžeme vidět například předpokládané propojení mezi bolestmi břicha (abdominal pain), průjmem (diarrhea) a nevolností (nausea).



Obrázek 8.6: Uzlový graf propojení symptomů

Kapitola 9

Závěr

Mým úkolem bylo navrhnout a implementovat architekturu schopnou analyzovat a vizualizovat RDF data. I přesto, že se v aplikaci EEA ElasticSearch RDF Indexer vyskytly chyby, přístup ke zdrojovému kódu aplikace tento problém však značně zmírňuje a nalezené chyby byly opraveny. Vytvoření uživatelského rozhraní na správu a monitorování indexů značně rozšířilo a usnadnilo použitelnost Indexeru. Serverová nadstavba s API rozhraním umožňuje integraci i s jinými například automatizovanými systémy. Propojení s produkty od společnosti Elastic tvoří z architektury velmi silný nástroj pro danou práci a proto si stojím za výběrem jednotlivých technologií. Myslím si, že úkol práce jsem splnil a projekt připravil na budoucí rozvoj.

Jak už bylo zmíněno v textu práce, EEA ElasticSearch RDF Indexer měl v sobě řadu chyb, bylo by tedy vhodné otestovat její celou funkcionalitu. Z testování efektivity také vyplynulo, že by bylo příhodné některé části procesu indexování optimalizovat. Naimplementované uživatelské rozhraní přenechává většinu validace na serverové části vytvořené aplikace. Doimplementování validace na frontend by usnadnilo práci a ujasnilo kritéria vyplnění jednotlivých polí. V pokročilé části konfigurace by uživateli pomohla lépe popsaná jednotlivá políčka k vyplnění. Uživatelské části také chybí jazykové mutace.

Literatura

[1] Apache INCREDIBLE. Apache Superset. https://superset.apache.org/, 2015. Navštíveno 5.11.2020, git: https://github.com/apache/incubator-superset/blob/master/INTHEWILD.md.

[2] M. Birbeck, B. Adida, I. Herman, and M. Sporny. RDFa 1.1 primer - third edition. Technical report, W3C, Mar. 2015. Navštíveno 5.11.2020 https://www.w3.org/TR/2015/NOTE-rdfa-primer-20150317/.

[3] D. Brickley and R. Guha. RDF schema 1.1. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[4] G. Carothers. RDF 1.1 n-quads. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-n-quads-20140225/.

[5] G. Carothers and A. Seaborne. RDF 1.1 trig. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-trig-20140225/.

[6] Chart.js. Chart.js. https://www.chartjs.org/, 2013. Navštíveno 5.11.2020, git: https://github.com/chartjs/Chart.js.

[7] D. Crockford. Json. https://www.json.org/. Navštíveno 02.01.2021.

[8] R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 concepts and abstract syntax. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[9] Elastic NV. elastic.co. https://www.elastic.co/, 2010. Navštíveno 10.12.2020.

[10] Elastic NV. Elasticsearch. https://www.elastic.co/elasticsearch/, 2010. Navštíveno 10.12.2020.

[11] Elastic NV. Kibana. https://www.elastic.co/kibana, 2014. Navštíveno 10.12.2020.

[12] R. T. Fielding. Rest. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Navštíveno 08.03.2021.

[13] I. Fonticons. Font awesome. https://fontawesome.com/how-to-use/on-the-web/using-with/react. Navštíveno 02.04.2021.

[14] F. Gandon and G. Schreiber. RDF 1.1 XML syntax. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/.

[15] R. Group. Recharts. https://recharts.org/. Navštíveno 12.04.2021 https://github.com/recharts/recharts.

[16] S. Harris and A. Seaborne. SPARQL 1.1 query language. Technical report, W3C, Mar. 2013. Navštíveno 19.11.2020 https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[17] P. Hayes. RDF semantics. Technical report, W3C, Feb. 2004. Navštíveno 5.11.2020 https://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[18] F. Inc. React. https://reactjs.org/. Navštíveno 02.01.2021.

[19] S. J. C. Jason Quense. React bootstrap. https://react-bootstrap.netlify.app/. Navštíveno 10.01.2021 https://github.com/react-bootstrap/react-bootstrap.

[20] T. A. Jr., F. Rivoal, and E. Etemad. CSS snapshot 2020. Technical report, W3C, Dec. 2020. Navštíveno 12.01.2021 https://www.w3.org/TR/2020/NOTE-css-2020-20201222/.

[21] G. Kellogg, M. Lanthaler, and M. Sporny. JSON-ld 1.0. Technical report, W3C, Jan. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-json-ld-20140116/.

[22] A. B. Laboratories. CRON. Technical report, AT&T, May 1975. Navštíveno 29.12.2020 https://pubs.opengroup.org/onlinepubs/007904975/utilities/crontab.html.

[23] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195, 2015. Navštíveno 03.01.2021.

[24] Matei Zaharia. Apache Spark. https://spark.apache.org/, 2014. Navštíveno 04.01.2021, git: https://github.com/apache/spark.

[25] N. U. Matt Zabriskie. Axios. https://axios-http.com/. Navštíveno 12.01.2021 https://github.com/axios/axios.

[26] Nadace Wikimedia. Wikidata.org. https://www.wikidata.org/, 2019. Navštíveno 26.04.2021.

[27] Oracle Corporation. Java. https://www.java.com/, 1995. Navštíveno 12.05.2021.

[28] Oracle Corporation. Java Runtime Environment. https://www.oracle.com/java/technologies/javase-jre8-downloads.html, 1995. Navštíveno 12.05.2021.

[29] E. Prud'hommeaux and G. Carothers. RDF 1.1 turtle. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-turtle-20140225/.

[30] SANSA. Sansa stack. http://sansa-stack.net/. Navštíveno 5.11.2020.

[31] A. Seaborne and G. Carothers. RDF 1.1 n-triples. Technical report, W3C, Feb. 2014. Navštíveno 5.11.2020 https://www.w3.org/TR/2014/REC-n-triples-20140225/.

[32] P. Software. Spring boot. https://spring.io/projects/spring-boot. Navštíveno 02.01.2021.

[33] M. Suignard and M. Duerst. IRI. Technical report, W3C, Jan. 2005. Navštíveno 12.05.2021 https://datatracker.ietf.org/doc/html/rfc3987.

[34] Z. Szabo. EEA ElasticSearch RDF Indexer. https://github.com/eea/eea.elasticsearch.river.rdf. Navštíveno 18.12.2020.

[35] Tableau Software. Tableau. https://www.tableau.com/, 2021. Navštíveno 5.11.2020.

[36] The Apache Software Foundation. Apache Flink. https://flink.apache.org/, 2013. Navštíveno 04.01.2021, git: https://github.com/apache/flink.

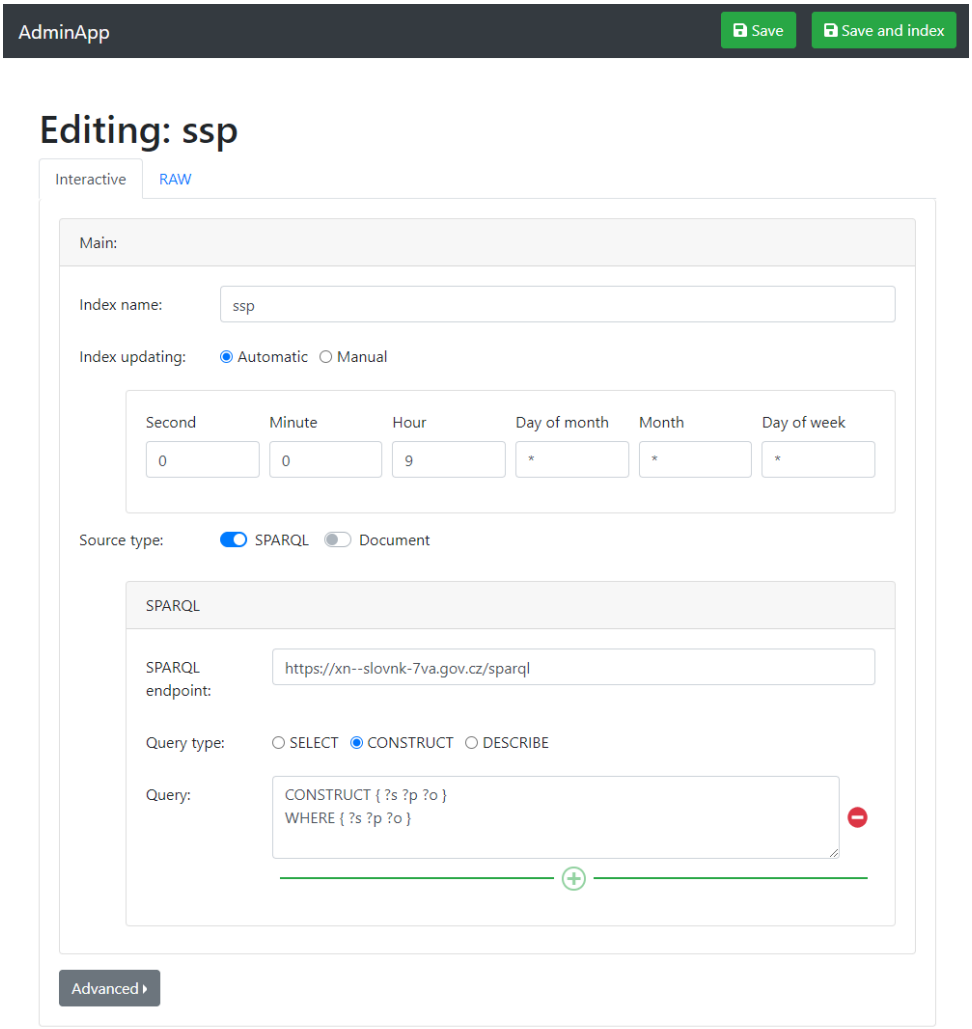
[37] A. van Kesteren and S. Ruby. URL. Technical report, W3C, Dec. 2016. Navštíveno 12.05.2021 https://www.w3.org/TR/2016/NOTE-url-1-20161206/.

[38] W3C. SPARQL 1.1 overview. Technical report, W3C, Mar. 2013. Navštíveno 5.11.2020 https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

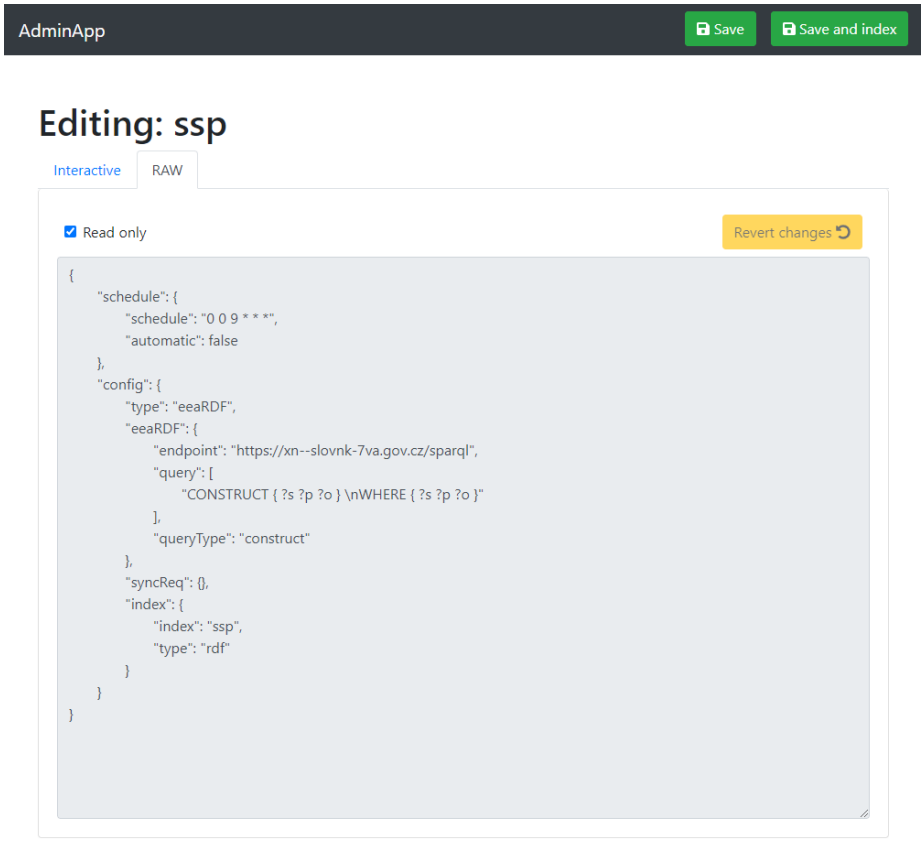
[39] E. Willigers, D. Storey, B. Brinza, D. Schulze, A. Bellamy-Royds, and C. Lilley. Scalable vector graphics (SVG) 2. Technical report, W3C, Oct. 2018. Navštíveno 12.01.2021 https://www.w3.org/TR/2018/CR-SVG2-20181004/.

Příloha A

Snímky obrazovky



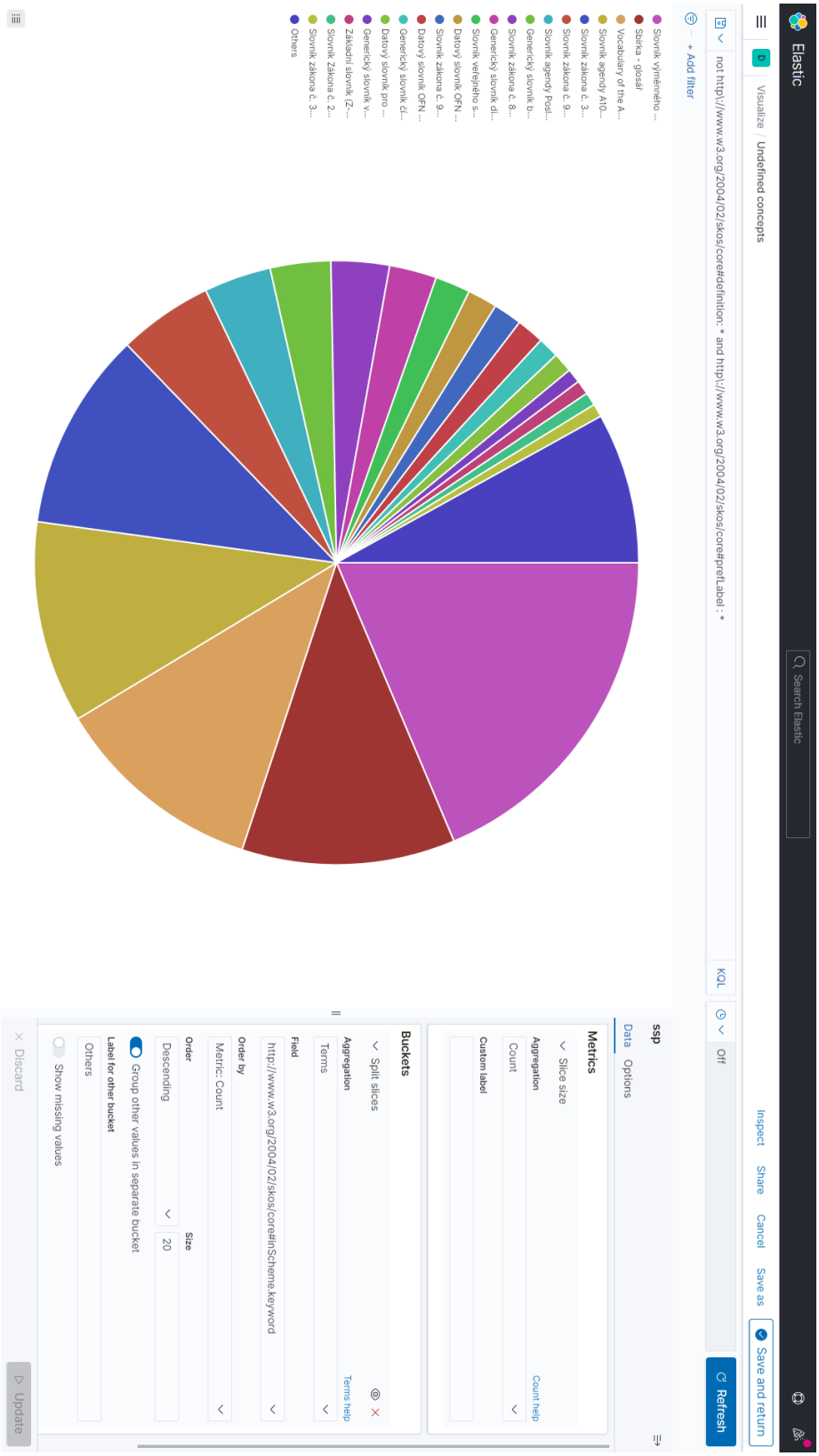
Obrázek A.1: Nastavení konfigurace SSP v administrátorské aplikaci



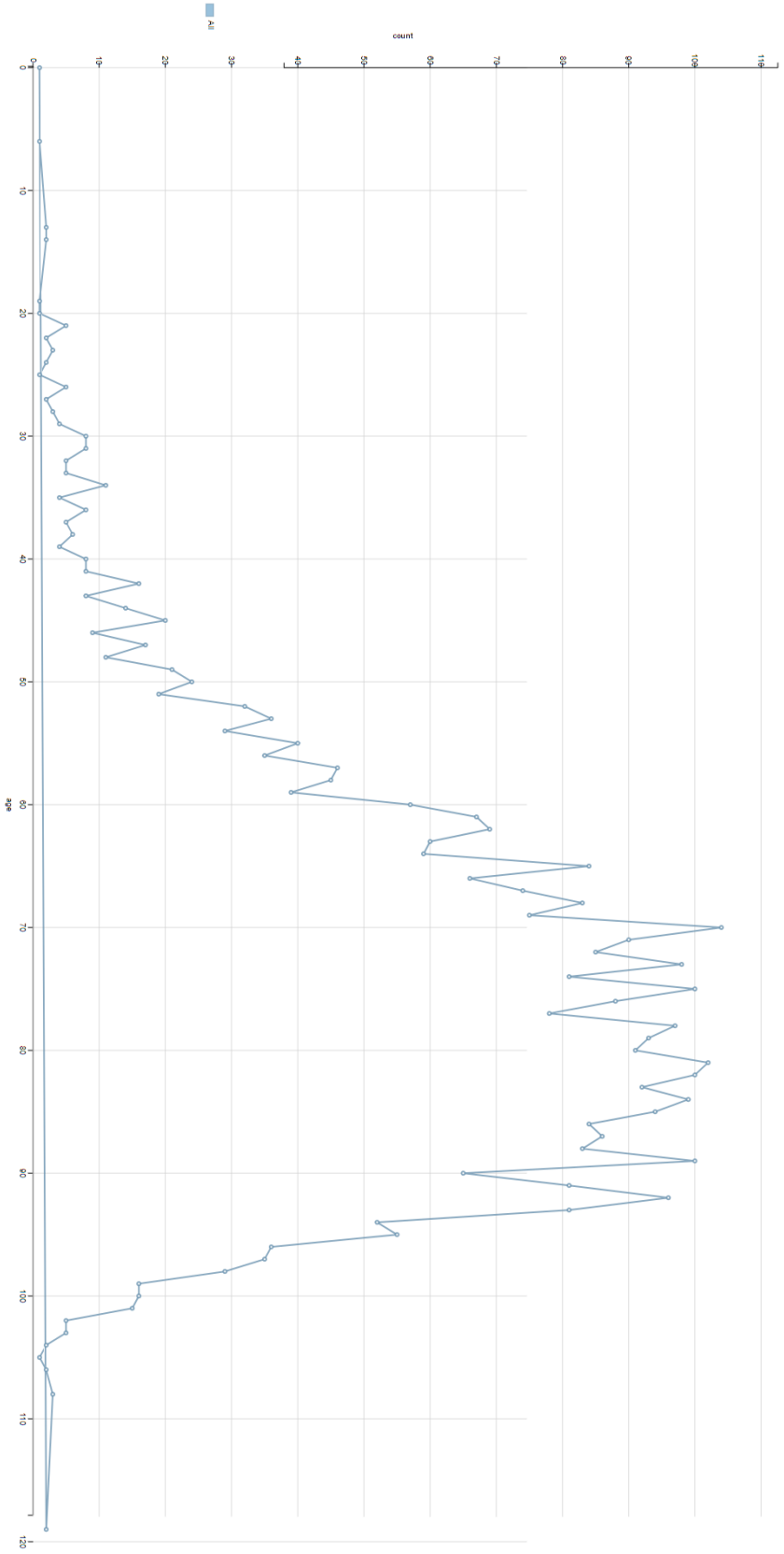
Obrázek A.2: Nastavení přímá konfigurace SSP v administrátorské aplikaci

Obrázek A.3: Přehled indexů v administrátorské aplikaci

Obrázek A.4: Procházení indexu v Kibaně



Obrázek A.5: Vytváření grafu v Kibaně



Obrázek A.6: Graf zaznamenaných úmrtí z Wikidata

Příloha B

Obsah elektronické přílohy

- aplikace.zip
  - BP - frontend — Frontendová část aplikace
    - .idea
    - docs — Dokumentace a modelování
    - src — Zdrojový kód aplikace
    - .gitignore
    - package.json
    - package-lock.json
    - react.md
    - README.md — Návod k instalaci
  - BP - server — API server s indexerem
    - .idea
    - docs — Dokumentace a modelování
    - frontend — Složka pro vložení frontendu
    - src — Zdrojový kód aplikace
    - .gitignore
    - crontab
    - docker-compose.yml — Nastavení pro vytvoření kontejnerů
    - Dockerfile — Nastavení pro vytvoření image aplikace
    - EEA_Indexer_README.rst — Původní popis aplikace Indexeru
    - eea-rdf-river-indexer.iml
    - eea-rdf-river-plugin.iml
    - LICENSE.md — Licence aplikace Indexeru
    - pom.xml
    - README.md — Návod k instalaci