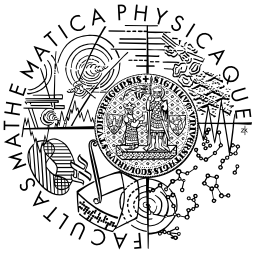


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Artur Finger

Webová aplikace pro prezentaci integrovaných výsledků kontrol

Katedra softwarového inženýrství

Vedoucí bakalářské práce:	Doc. Mgr. Martin Nečaský, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové a datové inženýrství

Praha 2016

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

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 Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Webová aplikace pro prezentaci integrovaných výsledků kontrol

Autor: Artur Finger

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: Doc. Mgr. Martin Nečaský, Ph.D., Katedra softwarového inženýrství

Abstrakt: V dnešní době vzniká iniciativa publikovat data státních orgánů jako Linked Data, která se spojí s ostatními souvisejícími daty sémantického webu. Dohromady pak vzniká Linked Data Cloud, který slouží jako bohatý datový zdroj pro řadu aplikací. Nám se i přes nečekané potíže, podařilo do této podoby převést data o proběhlých kontrolách Státní veterinární správy a to čistě za použití ETL nástroje UnifiedViews. K popisu dat jsme použili známé RDF slovníky. Posléze, také pomocí UV, jsme data SVS integrovali s Linked Daty kontrol České obchodní inspekce. Nad SPARQL databází integrovaných dat jsme vybudovali responzivní webovou aplikaci, určenou k vyhledávání poctivých podnikatelů a jejich provozoven. S daty se pracovalo exkluzivně ve formátu RDF a za použití databázového serveru Virtuoso se podařilo naimplementovat i vyhledávání pomocí souřadnic. Aplikace je připravená pro další datové zdroje a vytěžená data SVS jsou otevřeně dostupná.

Klíčová slova: Linked Data semantický web RDF ontologie výsledky kontrol

Title: Web application for presentation of integrated inspection data

Author: Artur Finger

Department: Department of Software Engineering

Supervisor: Doc. Mgr. Martin Nečaský, Ph.D., Department of Software Engineering

Abstract: An initiative arises across the globe, to publish government data as Linked Data, thus connecting it to other parts of the semantic web. Together the data forms the Linked Data Cloud, which serves as a rich data source for numerous applications. Overcoming unexpected problems, we managed to convert data about inspections carried out by the State Veterinary Administration into that format. For that we used an ETL tool called UnifiedViews. Resulting data were described using well-known RDF ontologies. Then, also using UV, we integrated this data with Linked Data of the Czech Trade Inspection Authority. We created a web application, that uses this integrated database to search for fair businesses. Using a database server called Virtuoso, we managed to implement search by geographical coordinates. Our application is extensible by new data sources and extracted SVA data are available as open Linked Data.

Keywords: Linked Data semantic web RDF ontology inspection data

Děkuji doc. Mgr. Martinu Nečaskému, Ph.D. za vedení práce a Mrg. Petru Škodovi za školení a pomoc s nástrojem UnifiedViews.

Obsah

Pojmy

- **RDF** - Je to jednoduchý datový formát (viz W3C, 2014), na kterém je založený veškerý sémantický web. Ve zkratce funguje tak, že libovolná data se ukládají jako trojice (subjekt, predikát, objekt). Subjekt a predikát jsou URI a objekt je buď také URI nebo otypovaný text (typ textové hodnoty se specifikuje pomocí datových typů xsd). Množina trojic vytváří orientovaný graf, jenž může být pojmenovaný. Na pojmenování grafů se také používají URI. Množina grafů se nazývá datová sada (*angl. dataset).* Mezi grafem a datovou sadou nebudeme ostře rozlišovat, protože jeden graf lze také počítat jako datovou sadu. Veřejně dostupný SPARQL server, který hostuje několik grafů, se nazývá přístupový bod (*angl. sparql endpoint).* Podstatou RDF a sémantického webu je jednoduché propojování grafů, ať už jsou kdekoliv.

```
<http://priklad.cz/uzel/dellInspiron15R>
        a <http://znamySlovnikNotebooku/Notebook> .

<http://priklad.cz/uzel/dellInspiron15R>
        <http://znamySlovnikNotebooku/maRAMvGB>
                "4"^^<http://www.w3.org/2001/XMLSchema#integer> .

#zavademe zkratky
zs : http://znamySlovnikNotebooku/
pr : http://priklad.cz/uzel/
```

  Uvedený příklad obsahuje dvě trojice. Máme uzel typu *zs:Notebook*. Uzel má vlastnost *zs:maRAMvGB* s hodnotou "4", datového typu *xsd:integer*. Pomocí trojic (když počítáme datový typ hodnoty, tak čtveřic) lze v RDF popsat cokoliv. Vlastnost je jen jiné pojmenování pro predikát. Objekt může být buď literál (neboli otypovaný text) nebo URI. Vlastnost *a* je jen zkratkou za predikát http://www.w3.org/1999/02/22−rdf−syntax−ns#type, který se používá k udávání typu subjektu. V našem příkladu jsme uvedli subjekt *pr:dellInspiron15R* typu *zs:Notebook*.

- **RDF graf** - RDF graf je pojmenovaná množina RDF dat. RDF jsou grafová data a k pojmenování grafů se vždy používá URI/IRI.

- **RDF ontologie/slovníky** - Pro přirovnání se zamysleme nad technologií XML. V XML je možné si zvolit libovolné názvy značek (*angl. tag),* ale jednoduše se stane, že nikdo kromě autora jim nerozumí. Speciálně to nastane pokud jsou značky v češtině, ale čtenář je cizinec. Na druhou stranu, HTML všichni dobře rozumí. HTML je podmnožina XML a vtip je v tom, že používá jen několik dobře zavedených, zdokumentovaných a dlouho známých značek.
  Podobně pro RDF existují slovníky, což jsou jen seznamy URI, která mají dobře specifikovaný význam. Označíme-li něco například `http://schema.org/Book`, všichni RDF programátoři budou vědět, že se jedná o knihu. Je to díky tomu, že schema.org je už dost známé.
  Obecně to ovšem s RDF slovníky není tak úplně jednoduché. Specifikace HTML je jen jedna, ale v RDF je slovníků mnoho a stále přibývají nové.

Každý slovník popisuje kousek reálného světa - jeden knihy, druhý rezervace, třetí adresy. Navíc slovníky nejsou disjunktní. Velmi často se stane, že více slovníků popisuje to samé (např. datum) nebo naopak neexistuje slovník, který by popsal něco specifického. V neposlední řadě je potřeba zmínit, že žádné RDF slovníky ještě nedosáhly dostatečné důležitosti, aby se staly webovými standardy. Při navrhování nové ontologie pro otevřená data (popisování nějakých nových dat v RDF) je tedy potřeba co nejčetněji používat známé slovníky, aby se zvýšila pravděpodobnost, že datům porozumí jejich uživatelé.

- **SPARQL** - Jedná se o dotazovací jazyk pro RDF (viz Harris a kol., 2013). SPARQL je pro RDF to, čím XQuery je pro XML, nebo SQL pro relační tabulky. SPARQL se hodně inspiruje SQL, má podobnou syntaxi a je deklarativní. Podobně jako SQL, SPARQL dotazy umožňují data zobrazovat, mazat i vytvářet. Na druhou stranu se SPARQL od SQL liší právě kvůli datovému formátu, nad kterým se dotazuje - proto ve SPARQLu například neexistují JOINY. Celkově se SPARQL sice podobá SQL syntakticky, ale sémantika a způsob vyhodnocování jeho dotazů jsou dost odlišné. SPARQL dotaz se často týká více grafů současně, této množině grafů se pak říká datová sada.

```
PREFIX s: <http://schema.org/>
PREFIX t: <http://vypod.cz/test/>
CONSTRUCT {
        t:informace a t:DebugovaciInformace ;
                t:pocet_kontrol ?pocet_kontrol .
}
FROM <http://vypod.cz/coi>
WHERE {
        {
                SELECT (COUNT(*) AS ?pocet_kontrol)
                WHERE {
                        ?kontrola a s:CheckAction .
                }
        }
}
```

Uvedený příklad je celkem pokročilý SPARQL dotaz. Tento dotaz v grafu http://vypod.cz/coi spočítá počet kontrol (*s:CheckAction*). Kontroly se počítají v pod-dotazu *SELECT*. Jako výsledek se vytvoří dvě trojice popisující ladící informace. Mezi další důležité SPARQL dotazy patří *SELECT*, *CONSTRUCT* a *DESCRIBE*.

Úvod

0.1 Kontext

Sémantický web roste a v Česku vzniká iniciativa zpřístupňovat veřejně dostupná data programátorům v podobě Linked Data. Hlavním iniciátorem otevírání/zpřístupňování veřejných dat státní správy České republiky je organizace Opendata.cz (viz Charles University in Prague, 2016), která zároveň provozuje server `http://linked.opendata.cz`, kde se otevřená data (*angl. open data)* uchovávají a jsou přístupná veřejnosti. Opendata.cz otevírá data programátorům a tím pádem i široké veřejnosti, aby s nimi mohli libovolně pracovat (viz Cibulka, 2015). Tato organizace zpřístupňuje otevřená data ve formátu RDF pomocí SPARQL serveru. Mezi otevřená data patří mimo jiné informace z živnostenského rejstříku a záznamy výsledků kontrol státních kontrolních orgánů, například České obchodní inspekce. Data v tomto otevřeném formátu jsou sice přístupná veřejnosti, ale rozhraním je dotazovací jazyk SPARQL (podobný SQL), takže se k nim ve výsledku dostane jen programátor, ovšem i jemu zabere nějaký čas z dat něco zajímavého vyčíst. Proto je vhodné nad otevřenými daty v RDF vybudovat nějaké zajímavé webové nebo mobilní aplikace, které může snadno využít neprogramátorská veřejnost, čímž otevřenost dat nabude svého pravého významu. Budování těchto aplikací navíc podněcuje státní orgány publikovat svá data otevřeně, protože práce, kterou na to vynaloží, přinese něco užitečného všem.

Podstatou otevřených dat, včetně těch na doméně Opendata.cz, není budovat aplikace pro koncové uživatele, ale data propojovat, publikovat a zpřístupňovat odborníkům, kteří pak budou mít možnost data využívat v libovolné aplikaci ať už pro dobročinné nebo komerční účely (v závislosti na licenci).

Na druhou stranu zdaleka ne všechny státní orgány, které produkují data, k nimž by veřejnost měla mít přístup, je otevřely pro programátory ve formě RDF. Jedním takovým orgánem je Státní veterinární správa (SVS), která nad svou soukromou databází poskytuje vlastní webovou aplikaci. Nikdo jiný než SVS nemůže její data využívat k programování a propojování souvislostí.

0.2 Cíle

Cílem této práce bylo:

1. Vytvořit integrovanou databázi výsledků kontrol, které provádějí instituce veřejné správy (např. Česká obchodní inspekce). Měly být vybrány 2 různé instituce, jejichž výsledky kontrol jsou veřejně dostupné.

2. Analyzovat datové modely výsledků kontrol vybraných institucí a navrhnout ontologii, která by tyto datové modely sjednotila. Ontologie měla být postavena dle principů Linked Data (viz Linked Data community a Heath, 2016), tj. především měla vhodně kombinovat a doplňovat již existující ontologie a slovníky (např. slovník Schema.org (viz schema.org community, 2016)).

3. Automatizovaně vytěžovat data o výsledcích kontrol (poskytovaná vybranými institucemi) z jejich webových stránek pomocí nástroje UnifiedViews (UV) (viz Charles University in Prague a EEA, 2016), převádět je do podoby Linked Data dle navržené ontologie a publikovat je na doméně Opendata.cz.

4. Vyvinout webovou aplikaci, která by s integrovanými daty umožnila pracovat koncovým uživatelům.

Je potřeba podotknout, že ontologie v druhém cíli a integrovaná databáze v prvním cíli se netýkají toho samého. Navržené ontologie v druhém cíli mají vypovídat o tom, jaký bude model vytěžených dat ve formátu RDF, která mají být publikována otevřeně na doméně Opendata.cz pro využití jinými programátory. Zatímco integrovaná databáze už se nemusela tolik držet principů Linked Data, protože není určená pro veřejné využití, ale spíše pro potřeby webové aplikace ze čtvrtého cíle. Stačí, aby integrovaná databáze nakonec obsahovala jen podmnožinu vytěžených dat. Musí především obsahovat ty atributy, které data obou institucí mají společné.

0.3 Struktura práce

Některé cíle mají mnoho společného a často jeden krok pomohl k dosažení více cílů. Proto se u různých cílů jednotlivé body někdy opakují.

První cíl vyžadoval:

- výběr institucí
- rozmyšlení požadavků na integraci na základě konceptuálních modelů datových zdrojů
- analýza logického modelu dat ČOI a navržené ontologie dat SVS
- výběr typu databáze
- návrh logického modelu integrované databáze
- návrh těžení a integrace dat
- vytěžení dat ČOI a SVS
- integrace dat
- testování integrace

Druhý cíl vyžadoval:

- analýzu konceptuálního modelu dat SVS
- analýzu konceptuálního a logického modelu dat ČOI
- vytvoření ontologie pro data SVS s ohledem na ontologii ČOI s použitím známých slovníků

Třetí cíl vyžadoval:

- rozmyšlení požadavků na vytěžování dat vybraných institucí
- návrh vytěžování dat
- zaučení s nástrojem UV

- vytvoření znovupoužitelných datovodů na vytěžování dat SVS a stažení dat ČOI
- testování datovodů

Čtvrtý cíl vyžadoval:

- rozmyšlení funkčních a kvalitativních požadavků na webovou aplikaci
- vytvoření konceptuálního modelu integrované databáze
- návrh funkcionality webové aplikace
- návrh GUI
- volba software pro databázi (a tudíž i konceptuálního modelu integrované databáze)
- vytvoření integrované databáze
- volba webových technologií
- implementace webové aplikace
- testování webové aplikace

0.4 Přehled kapitol

První kapitola se zabývá výběrem kontrolních institucí a následnou analýzou jejich dat. Ná základě toho se určí požadavky na integraci a vytěžování dat. Nakonec se také určí požadavky na funkcionalitu webové aplikace. Druhá kapitola popisuje návrh přesné podoby vytěžovaných a integrovaných dat. Dále probírá způsob vytěžování a integrace dat (nezávisle na implementaci). Nakonec se také zabývá návrhem GUI webové aplikace a scénáři jejího použití. Třetí kapitola pojednává o volbě použitých technologií pro implementaci vytěžování dat, tvorbu integrované databáze a tvorbu webové aplikace. Dále obsahuje implementační detaily vytěžování dat pomocí UV a implementační detaily webové aplikace. V neposlední řadě také popisuje způsob integrace nového datového zdroje. Čtvrtá kapitola objasňuje použité způsoby testování a výsledky testů. Testovalo se jak vytěžování dat, tak i jejich integrace a také funkcionalita webové aplikace. Pátá kapitola stručně uvádí možnosti použití webové aplikace. Závěr shrnuje průběh práce a možnosti budoucího rozšíření.

1. Analýza

Na začátku práce bylo nutné rozumně vybrat dvě instituce, jejichž data by sloužila jako zdroje naší integrované databáze. Posléze se zmapovala podobná díla a prozkoumal se obsah datových zdrojů vybraných institucí. Na základě toho jsme vznesli požadavky na obsah integrované databáze, podle nichž se později ubíral její vývoj. Dále jsme zavedli požadavky na vytěžování dat z vybraných datových zdrojů a také na funkcionalitu webové aplikace.

1.1 Výběr institucí

V České republice platí zákon o kontrole, přesněji *Předpis č. 255/2012 Sb.; Zákon o kontrole (kontrolní řád)*, na základě kterého fungují všechny kontrolní orgány veřejné správy. Mezi nejvýznamnější takové orgány patří Česká obchodní inspekce (ČOI), Nejvyšší kontrolní úřad (NKÚ), Český telekomunikační úřad (ČTÚ), Státní zemědělská a potravinářská inspekce (SZPI) a Státní veterinární správa (SVS). Každý kontroluje něco trochu jiného a každý publikuje informace o kontrolách jiným způsobem. Z nich pouze data ČOI a NKÚ jsou dostupná ve formátu RDF. ČOI publikuje data v RDF a data NKÚ jsou převedená do tohoto formátu díky Opendata.cz.

ČOI kontroluje *"právnické a fyzické osoby prodávající nebo dodávající výrobky a zboží na vnitřní trh, poskytující služby nebo vyvíjející jinou podobnou činnost na vnitřním trhu, poskytující spotřebitelský úvěr nebo provozující tržiště (tržnice), pokud podle zvláštních právních předpisů nevykonává tento dozor jiný správní úřad. ... Česká obchodní inspekce kontroluje: - dodržování podmínek stanovených k zabezpečení jakosti zboží nebo výrobků (kromě potravin) včetně jejich zdravotní nezávadnosti, podmínek pro skladování a dopravu; - zda se při prodeji zboží používají ověřená měřidla (pokud ověření podléhají), a zda používaná měřidla odpovídají příslušným předpisům, technickým normám či patřičnému schválení; - dodržování podmínek stanovených právními a jinými příslušnými předpisy pro poskytování určitých služeb a provozování některých specifických činností; - zda při uvádění výrobků na trh byly tyto výrobky opatřeny náležitým povinným označením, popřípadě zda k nim byl vydán či přiložen předepsaný certifikát, zda vlastnosti stanovených výrobků uvedených na trh odpovídají příslušným technickým požadavkům a podobně; - zda výrobky uváděné na trh jsou bezpečné; - zda jsou při sjednávání spotřebitelského úvěru dodržovány povinnosti stanovené právními předpisy (pokud dozor v daném případě nevykonává Česká národní banka)."* (viz Česká obchodní inspekce, 2016) ČOI publikuje data o kontrolách ve formátu RDF jako Linked Data.

NKÚ kontroluje vysoce postavené orgány, především ministerstva. *"Nejvyšší kontrolní úřad je nezávislý orgán, který vykonává kontrolu hospodaření se státním majetkem a plnění státního rozpočtu. Postavení, působnost, organizační strukturu a činnost upravuje zákon č. 1/1993 Sb., Ústava České republiky, hlava V., článek 97, a zákon č. 166/1993 Sb., o Nejvyšším kontrolním úřadu, ve znění pozdějších předpisů."* (viz Nejvyšší kontrolní úřad, 2016) Data o kontrolách NKÚ jsou dostupná ve formátu RDF díky Opendata.cz, která je získává z oficiálního zdroje ve formátu CSV a převádí je do RDF.

ČTÚ kontroluje hlavně dodržování zákonů při elektronických komunikacích. *"Český telekomunikační úřad jako národní regulátor služeb elektronických komunikací a poštovních služeb je zákonem zmocněn rozhodovat tyto druhy sporů: -spory mezi osobami vykonávajícími komunikační činnosti (§ 127 ZEK); -účastnické spory (§ 129 ZEK); -řízení o námitkách proti vyřízení reklamace; -řízení o námitkách proti vyřízení reklamace podle zákona o poštovních službách."* (viz Český telekomunikační úřad, 2016) ČTÚ publikuje data o výsledcích jejich kontrol pomocí webového publikačního systému 1.1, z nějž by bylo nemožné data vytěžit (publikační systém sice nabízí tlačítko *Stáhnout*, ale data takto stáhnout nelze kvůli nedostatku oprávnění). Ovšem po pozornější inspekci zdrojového kódu lze dohledat přímý odkaz na export z databáze ve formátu CSV. Tímto skrytým způsobem by data měla být přístupná jednoduše.

SZPI kontroluje převážně potraviny a pokrmy. *"SZPI je organizační složka státu, která je přímo podřízená ministerstvu zemědělství. Je orgánem státního dozoru zejména nad bezpečností, jakostí a řádným označováním potravin. ... SZPI kontroluje, v rámci stanovených kompetencí, zemědělské výrobky, potraviny nebo tabákové výrobky. Nově od roku 2015 přibyla do kompetencí SZPI také kontrola pokrmů v zařízeních společného stravování. Tyto kompetence se vztahují na výrobu, uchování, přepravu i prodej (včetně dovozu)."* (viz Státní zemědělská a potravinářská inspekce, 2015) SZPI publikuje data o kontrolách jako součást HTML stránek ve třech sekcích, zvaných *Potraviny na pranýři*, *Tematické kontroly* 1.2 a *Uzavřené provozovny*. Ze všech tří by se daly těžit informace ohledně kontrol malých i velkých podnikatelů, týkající se kvality prodávaných nebo používaných potravin.

SVS kontroluje především farmy, rybníky, mrazírny, sklady potravin atd. *"Státní veterinární správa (SVS) je organizací, která ze zákona vykonává dozor nad zdravím zvířat, nad tím, aby nebyla týrána, nad zdravotní nezávadností potravin živočišného původu ..."* (viz Státní veterinární správa, 2016a) SVS publikuje data o výsledcích kontrol prostřednictvím webové aplikace 1.4, která na mapě zobrazuje kontrolované provozovny a detaily o zjištěných závadách. Detailnější data o podnikatelích publikuje zvlášť v registračních tabulkách 1.3.

Při výběru kontrolních orgánů jsme hledali dva, které se k sobě nejlépe hodí. Jejich publikovaná data o kontrolách by měla mít podobnou strukturu a co nejvíce společných atributů. Proto jsme zavrhli NKÚ, jelikož jeho hlavním posláním je kontrolovat ministerstva, zatímco všechny ostatní zmíněné orgány kontrolují poněkud menší podnikatele. SZPI jsme zavrhli kvůli neúplnosti dat, která zveřejňuje. V každé sekci se našel nějaký nedostatek. Hlavní nedostatek v sekci *Potraviny na pranýři* je ten, že se publikují pouze závadné kontroly, nikoliv nezávadné. Nešlo by tedy posoudit, zda je nějaká provozovna v pořádku nebo zda prostě zatím nebyla kontrolovaná. V sekci *Tematické kontroly* se zase neuvádí IČ při nezávadných kontrolách (a závadných kontrol je poměrně dost málo). To vede k podobnému problému jako tomu bylo v předchozí sekci. Sekce *Uzavřené provozovny* už se netýká tolik kontrol, jako spíš jednorázového uzavírání provozoven, kde je stav potravin a hygienický standard nepřípustný. Zbyly ČOI, SVS a ČTÚ. Všechny tři orgány mají podobnou strukturu dat, zabývají se podnikateli, poskytují informace o tom, kde kontrola proběhla, jaké byly nalezené závady, jaká byla výše udělené pokuty atd. Nakonec jsme zvolili SVS a ČOI díky větší tématické podobnosti. ČOI kontroluje převážně kvalitu a bezpečnost zboží a SVS hlavně

Obrázek 1.1: Záznamy o kontrolách ČTÚ

	Název výrobku	DP / DMT / Čís	Místo kontroly	Výrobce
>	Vyškovský Březňák	02.01.16	CZECH BEVERAGE INDUSTRY COMPANY a.s. Vyškov (náměstí Čsl. armády 116/4, 68201 Vyškov)	CZECH BEVERAGE INDUSTRY COMPANY a.s., Na Valentince 644/15, 150 00 Praha, Provoz: Pivovar Vyškov Nám. Čsl. armády 116/4, 682 01 Vyškov
>	Vyškovský Džbán	02.01.16	CZECH BEVERAGE INDUSTRY COMPANY a.s. Vyškov (náměstí Čsl. armády 116/4, 68201 Vyškov)	CZECH BEVERAGE INDUSTRY COMPANY a.s., Na Valentince 644/15, 150 00 Praha, Provoz: Pivovar Vyškov Nám. Čsl. armády 116/4, 682 01 Vyškov
>	Nakládaný camembert gl	27.08.15	Procházka a.s. Roudnice nad Labem 627, 41301 Roudnice nad Labem)	Procházka a.s., Chelčického 627, 41301 Roudnice nad Labem
>	Bramborák s uzeným ma 5x150g	18.08.2015	Procházka a.s. Roudnice nad Labem (Chelčického 627, 41301 Roudnice nad Labem)	Procházka a.s., Chelčického 627, 41301 Roudnice nad Labem
>	Nakládaný camembert	27.08.15	Procházka a.s. Roudnice nad Labem (Chelčického 627, 41301 Roudnice nad Labem)	Procházka a.s., Chelčického 627, 41301 Roudnice nad Labem
>	BUDAPEŠŤSKÁ POMAZ	20.08.15	Smetanová cukrárna a.s. Praha 4 (Metodějova 1464, 14900 Praha 4)	Smetanová cukrárna a.s., Metodějova 1464, 14900 Praha 4
>	HENRI VEJCE	16.08.15	Smetanová cukrárna a.s. Praha 4 (Metodějova 1464, 14900 Praha 4)	Smetanová cukrárna a.s., Metodějova 1464, 14900 Praha 4

Obrázek 1.2: Výsledky tématických kontrol SZSPI

Veřejné zakáz eska SVS Tiskový servis Legislativa Kontakty E-podatelna Vý

Státní veterinární správa registrované subjekty

eAGRI ▸

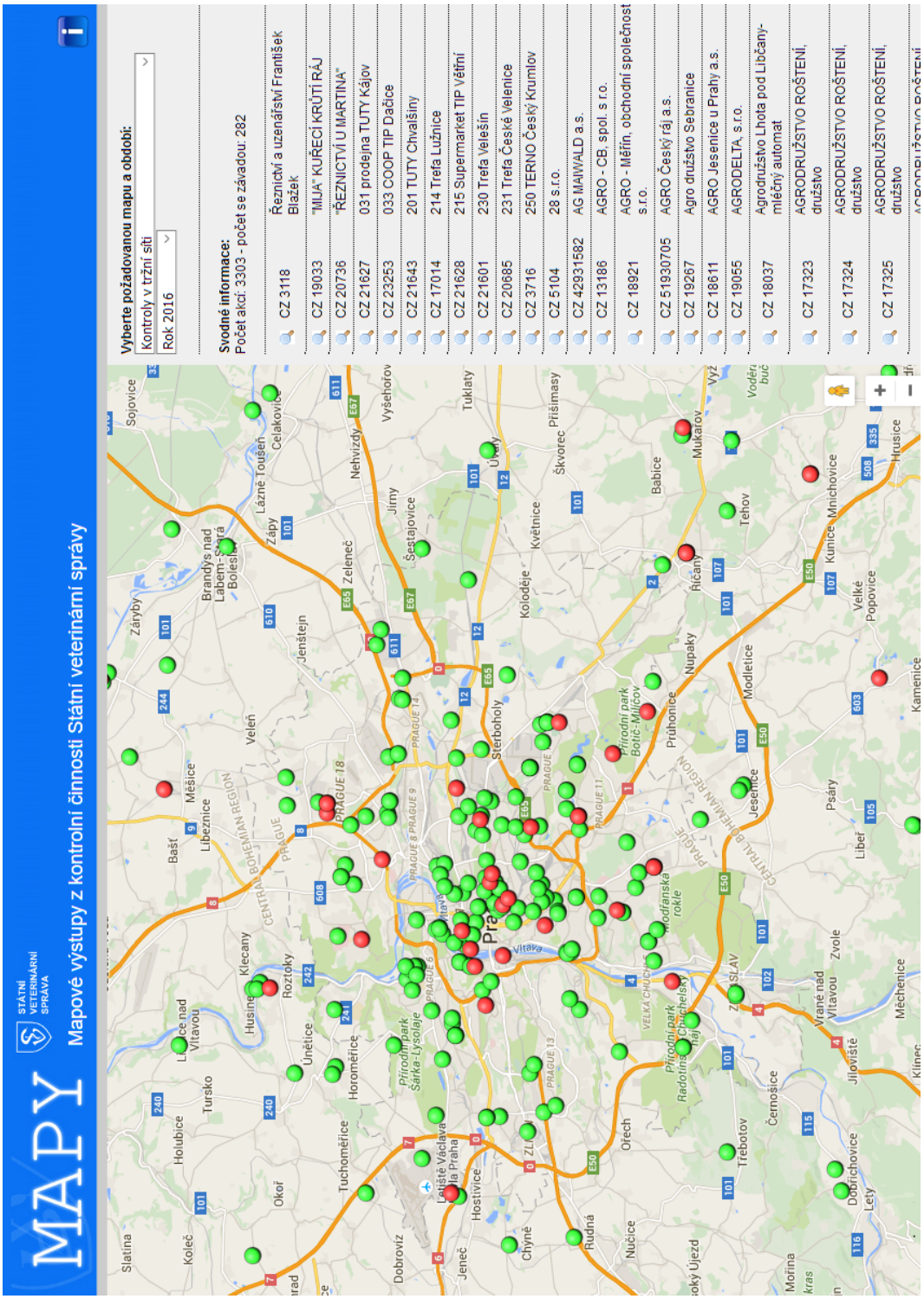
Hledaný výraz

átní veterinárr gistrované subjekty >

Zpracovatelé živočišných produktů schváleni a registrovaní pro obchod v rámci EU

registrační č	název	ulice a č. domu	PSČ	obec	typ závodu	datum sch
CZ 01	Mlékárna Benešov	Konopištská 905	25601	Benešov	Mlékárny	18.11.2003
CZ 03	MADETA a. s., závod Jindřichův Hradec	Jiráskovo předměstí 638	37701	Jindřichův Hradec	Mlékárny	07.10.2010
CZ 04	Mlékárna Klatovy a.s.	Za Tratí 640	33901	Klatovy	Mlékárny	24.11.2003
CZ 05	Eligo a.s., odštěpný závod Brno	Hněvkovského 603	61700	Brno	Mlékárny	11.11.2003
CZ 06	ALL-IMPEX, a.s.	Dělnická 548	53003	Pardubice	Mlékárny	19.11.2003
CZ 07	Mlékárna Hlinsko, a.s.	Kouty 53	53901	Hlinsko	Mlékárny	29.12.2003
CZ 08	OLMA a.s.Olomouc	Pavelkova 597	77900	Olomouc	Mlékárny	31.12.2003
CZ 09	Eligo a.s.	Třídvorská 712	28002	Kolín	Mlékárny	22.12.2003
CZ 10	Eurosérum,s.r.o.	Stříbro	34901	Stříbro	Mlékárny	05.11.2003
CZ 10038	POTRAVINY VYSOČINA s.r.o.	Český Dvůr 123	58001	Knyk	Čerstvé maso - jatky	19.12.2003
CZ 10038	POTRAVINY VYSOČINA s.r.o.	Český Dvůr 123	58001	Knyk	Čerstvé maso - bourárny	23.12.2003
CZ 10038	POTRAVINY VYSOČINA s.r.o.	Český Dvůr 123	58001	Knyk	Čerstvé maso - chladírny a mrazírny	01.01.2012
CZ 10038	POTRAVINY VYSOČINA s.r.o.	Český Dvůr 123	58001	Knyk	Masné výrobny	07.06.2013
CZ 10038	POTRAVINY VYSOČINA s.r.o.	Český Dvůr 123	58001	Knyk	Mleté maso, masné polotovary	22.07.2010
CZ 10154	MAKRO Cash & Carry ČR s.r.o.	Hradec Králové	50332	Hradec Králové	Čerstvé maso - chladírny a mrazírny	01.06.2011
CZ 10177	Rybářství Chlumec nad Cidlinou, a.s.	Chlumec nad Cidlinou	50351	Chlumec nad Cidlinou	Zpracování ryb	12.12.2003
CZ 10365	PRANTL Masný průmysl s.r.o.- reexpedice Ostrava	Prosekovická 899	72400	Ostrava	Čerstvé maso - chladírny a mrazírny	03.01.2011
CZ 10366	JUNIOR WORLDCUP '94, spol. s r.o.	Na Desátém 2276	70200	Ostrava	Chladírny a mrazírny - ostatních živ. produktů	01.07.2014
CZ 10370	Karel Oslizlok - reexpediční sklad uzenin	Cihelní 1575	70200	Ostrava	Čerstvé maso - chladírny a mrazírny	01.07.2014
CZ 10370	Karel Oslizlok - reexpediční sklad uzenin	Cihelní 1575	70200	Ostrava	Čerstvé drůbeží maso - chladírny a mrazírny	01.07.2014
CZ 10370	Karel Oslizlok - reexpediční sklad uzenin	Cihelní 1575	70200	Ostrava	Chladírny a mrazírny - ostatních živ. produktů	01.07.2014
CZ 105	MASO UZENINY PÍSEK, a.s.	Samoty 1533	39701	Písek	Čerstvé maso - jatky	31.12.2003
CZ 105	MASO UZENINY PÍSEK, a.s.	Samoty 1533	39701	Písek	Čerstvé maso - bourárny	31.12.2003
CZ 105	MASO UZENINY PÍSEK, a.s.	Samoty 1533	39701	Písek	Čerstvé maso - chladírny a mrazírny	31.12.2003
CZ 105	MASO UZENINY PÍSEK, a.s.	Samoty 1533	39701	Písek	Masné výrobny	31.12.2003

Obrázek 1.3: Registrace podnikatelů u Státní veterinární správy



Obrázek 1.4: Záznamy o kontrolách ve webové aplikaci SVS

jakost a nezávadnost výroby potravin. Naproti tomu ČTÚ se převážně zabývá technickou problematikou dodržování zákonů při elektronické komunikaci.

1.1.1 Shrnutí

Pro tuto práci jsme vybrali Českou obchodní inspekci (ČOI) a Státní veterinární správu (SVS). Tyto orgány se k sobě tématicky hodí a strukturou svých publikovaných dat se podobají více než ostatní.

1.2 Podobná díla

Tato práce je především jedinečná tím, že jako první vytěžuje data SVS a otevřeně je publikuje ve formátu RDF na doméně Opendata.cz. To zatím nikdo nedělal a velmi pravděpodobně se o to zatím ani nikdo nepokusil.
Co se týká webové aplikace (která je jen částí této práce), mezi podobná díla patří Veterinární mapa, Inspekce českých restaurací, Mapové výstupy z kontrolní činnosti SVS, Výsledky kontrol a Léková encyklopedie. Všechny jsou to webové nebo mobilní aplikace, které buď používají podobná data jako my nebo využívají RDF databáze pro backend. Zmíněné aplikace a jejich srovnání s naší budou podrobněji probrané v následující kapitole.

- **Výsledky kontrol** - webová aplikace v beta verzi, která umožňuje vyhledávání kontrol ČOI podle všech důležitých atributů. Naše aplikace má za úkol ji nahradit a liší se od ní především tím, že se bude zaměřovat hlavně na podnikatele a jejich provozovny. Navíc jako datový zdroj použijeme SVS a přidáme funkcionalitu hledání na mapě.
  `www.vysledkykontrol.cz`

- **Mapové výstupy z kontrolní činnosti Státní veterinární správy** - webová aplikace, která zobrazuje výsledky kontrol SVS na mapě. Výhodu má v tom, že přistupuje přímo do soukromé databáze SVS, zatímco my budeme používat veřejnou datovou sadu, kterou ovšem budeme muset nejdříve vytvořit. Dále se odlišuje tím, že hledání v ní probíhá klikáním na mapu a nezobrazuje IČO vlastníka provozovny, zatímco naše aplikace bude zobrazovat i IČO vlastníka, kde to bude možné a navíc umožníme lepší hledání - podle IČ, názvu, polohy atd. Naše aplikace bude navíc jako datový zdroj používat výsledky kontrol ČOI.
  `http://eagri.cz/public/app/svs_pub/mapy_vk`

- **Veterinární mapa** - mobilní aplikace pro Android, umí více méně úplně to samé jako předešlá webová aplikace až na to, že na mapě také vykresluje oblasti zamořené nákazami. Používá tu samou databázi. Naše aplikace se od ní liší ze stejných důvodů jako od předešlé aplikace a přesto, že naše aplikace není mobilní, je designovaná responzivně pro pohodlné použití jak na počítačích, tak i na mobilech a tabletech.
  `https://play.google.com/store/apps/details?id=eu.aquasoft.vetmapa&hl=cs`

- **Inspekce českých restaurací, Comsode** - webová aplikace v beta verzi. Umožňuje vyhledávání dobrých českých restaurací na mapě. Jestli je restaurace dobrá se usuzuje podle toho, zda měla problémy při nedávné

kontrole ČOI. Vyhledávání v této aplikaci je vybudováno pomocí speciálního nástroje na vytváření vyhledávačů na základě grafických diagramů viz `http://www.spinque.com/`. Naše aplikace se bude lišit tím, že jako zdroj dat použije také SVS a kód pro vyhledávání nebude generovaný nástrojem, ale bude psaný ručně.
`http://devel.spinque.com/comsode/coi/cz/`

- **Léková encyklopedie** - webová aplikace na vyhledávání léků, jejich ingrediencí, interakcí mezi nimi atd. Obsahově se od naší aplikace liší, ale společné mají to, že obě pro backend používají data ve formátu RDF a Virtuoso SPARQL server.
`www.lekovaencyklopedie.cz`

Další odlišností od výše zmíněných aplikací je důraz na podnikatele, případně jejich provozovny, a jejich poctivost. Veterinární mapa se více zaměřuje na jednotlivé kontroly, zatímco naše aplikace je zaměřená na osoby s IČ a názvem, které se zabývají specifickými obory podnikání.

1.3 Analýza datových zdrojů

Po výběru datových zdrojů (kontrolních institucí) bylo potřeba přesně zjistit, jaké informace zveřejňují. Výsledkem mapování zdrojů byly diagramy jejich konceptuálních modelů. Na základě získaných poznatků byla později navržená integrovaná databáze.

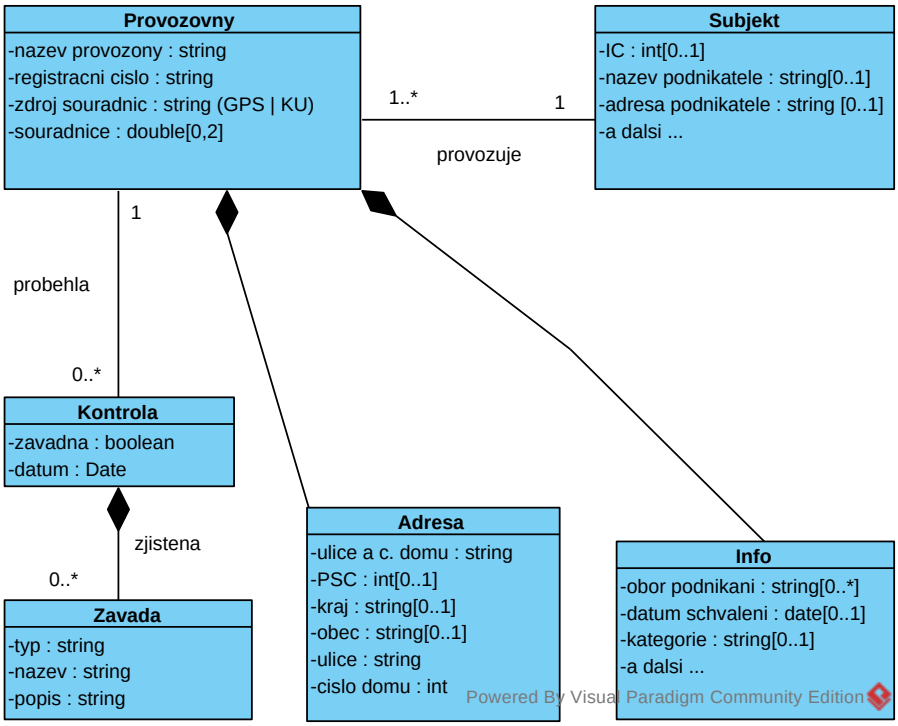
1.3.1 Konceptuální model kontrolních dat SVS

Konceptuální model dat SVS 1.5 jsme vytvořili na základě analýzy jejich webové aplikace a skupiny HTML tabulek s registracemi podnikatelů. Přitom se ovšem vynořily různé neočekávané obtíže. Například nešlo jednoznačně identifikovat jednotlivé kontroly. V datech se totiž rozlišují pouze závady. I když bylo jasné, že nějaké kontroly probíhají a právě díky nim byly závady zjištěné. Kontroly nebyly nijak odlišené, i když se na první pohled zdálo, že by mohly být. Analýza velkého množství dat však ukázala opak.

Tuto situaci jsme vyřešili přidáním předpokladu, že v jeden den nastala maximálně jedna kontrola. Tudíž všechny závady v konkrétní provozovně z jednoho dne pak patřily pod jednu unikátní kontrolu.

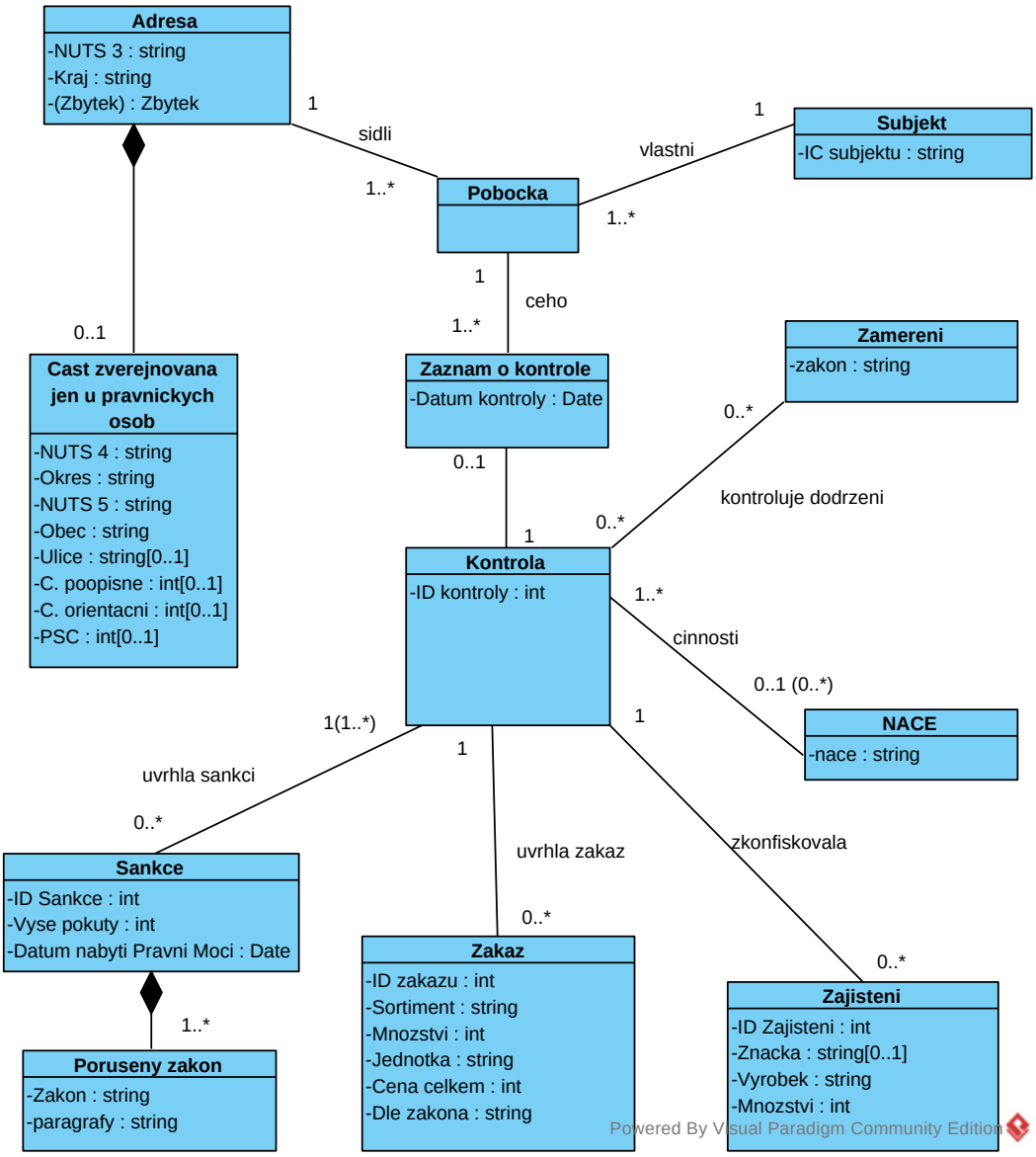
1.3.2 Konceptuální model kontrolních dat ČOI

Konceptuální model dat ČOI 1.6 jsme získali analýzou jejich veřejných exportů z databáze v CSV. V modelu je zajímavé, že prakticky jakákoli část může chybět - nejen jednotlivé atributy, ale i celé množiny tříd.
Po sepsání konceptuálního datového modelu ČOI ovšem nastal potenciální problém. ČOI začínalo plánovat změnu svého datového modelu a zavedení nového. Jak jsme po konzultaci s jeho autorem zjistili, důvodem byl záměr zavést obecnější model, který by měly začít používat i ostatní kontrolní instituce. Šlo o zavedení nového standardu. Naštěstí nový plánovaný model 1.7 byl nadmnožinou původního, takže šlo pracovat s původním.



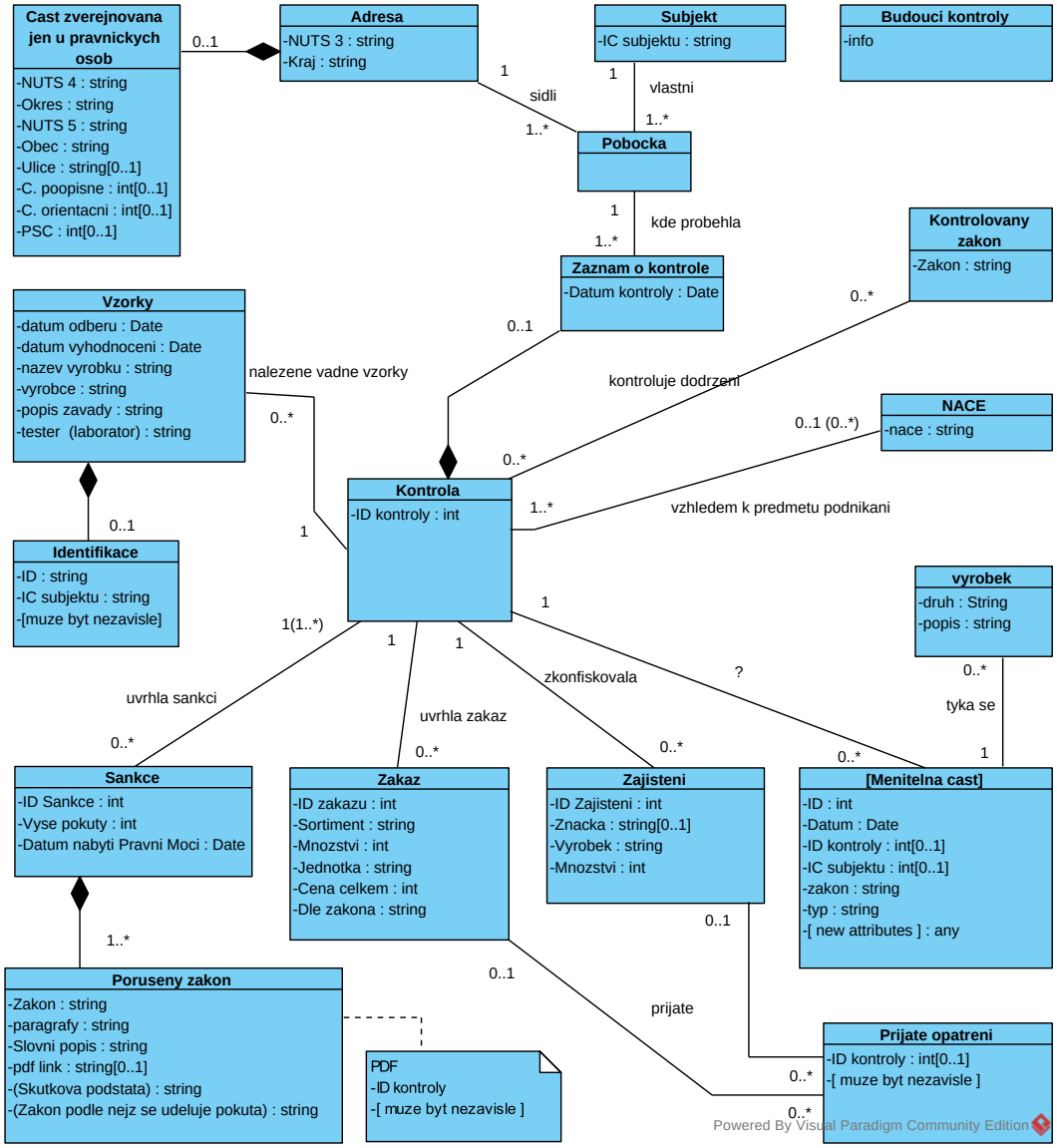
Obrázek 1.5: Konceptuální model kontrolních dat SVS

Třída *Provozovny* odpovídá reálným provozovnám v datech SVS a každou z nich je popisují atributy *název, registrační číslo, souřadnice a zdroj souřadnic*. Souřadnice jsou dvě reálná čísla, a nebo mohou chybět. V provozovně mohlo proběhnout libovolné (0 až *) množství kontrol. Na druhou stranu každá kontrola proběhla vždy v právě jedné provozovně. Kontrola mohla být závadná a má uvedené datum konání. Součást každé kontroly je libovolné množství závad, které byly zjištěny (jedna závada je součástí právě jedné kontroly). Podobně, součást každé provozovny je adresa a ta může nebo nemusí mít uvedené *psč, kraj, obec* a další atributy. Provozovny dále obsahují *Info* a každou provozuje právě jeden *Subjekt*. Naopak každý subjekt provozuje jednu nebo více provozoven.



Obrázek 1.6: Konceptuální model kontrolních dat ČOI

Třída *Subjekt* představuje podnikatele. Každý subjekt má uvedené IČO a vlastní jednu nebo více *Poboček* (provozoven). Pobočka patří právě jednomu subjektu, nachází se právě na jedné adrese a mohlo u ní být zaznamenaných jeden a více záznamů o kontrole. Záznamy o kontrole spojují data o kontrole s daty o pobočce a potažmo i o podnikateli. Každý záznam popisuje právě jednu kontrolu, ale ne každá kontrola má uvedený záznam. Kontroly zjistily libovolný počet *Sankcí, Zákazů a Zajištění*. Každá sankce obsahuje informaci o tom, který zákon byl porušen. Každá kontrola může mít uvedených libovolně mnoho *Zaměření* (zákon, jehož dodržení se kontrolovalo) a *NACE* (kód oboru podnikání dané provozovny). Každá adresa může nebo nemusí obsahovat volitelnou část, podle toho zda vlastníkem je právnická nebo fyzická osoba.



Obrázek 1.7: Plánovaný nový koceptuální model kontrolních dat ČOI

Platí vše z originálního modelu 1.6 a navíc u *Sankcí* je *Porušený zákon* detailněji popsaný včetně přiloženého oficiálního PDF o porušeném zákonu a sankci. Každý *Zákaz* a *Zajištění* může mít uvedených libovolně mnoho přijatých opatření. Navíc krom *Zákazů*, *Zajištění* a *Sankcí* může přibýt i další třída podobného rázu (*Měnitelná část*) a ta se bude týkat libovolného počtu *Výrobků*. Navíc se u každé kontroly bude uvádět libovolné množství vzorků a každý z nich bude obsahovat identifikační údaje.

1.3.3 Obecné problémy analýzy konceptuálních modelů

Konceptuální modely ČOI a SVS jsme odvozovali postupem reverzního inženýrství z podkladů jako jsou WWW stránky, soubory v CSV apod. Nic jiného nebylo k dispozici. Tato technika má ovšem základní problém, že se jedná jen o odhad. Nelze rozhodnout, zda nějaký atribut je povinný nebo ne. I když atribut nikde v dostupných datech nechybí, není vyloučeno, že v budoucnu, když vzniknou nová data, bude vynechán. Na druhou stranu, pokud atribut byl alespoň jednou vynechán, tak už je jasné, že není povinný. Přesto může být stále zajímavý a použitelný, pokud má dostatečně častý výskyt.
Zvolené konzervativní řešení při odhadování datových modelů a později při vývoji webové aplikace bylo předpokládat, že cokoliv může chybět.

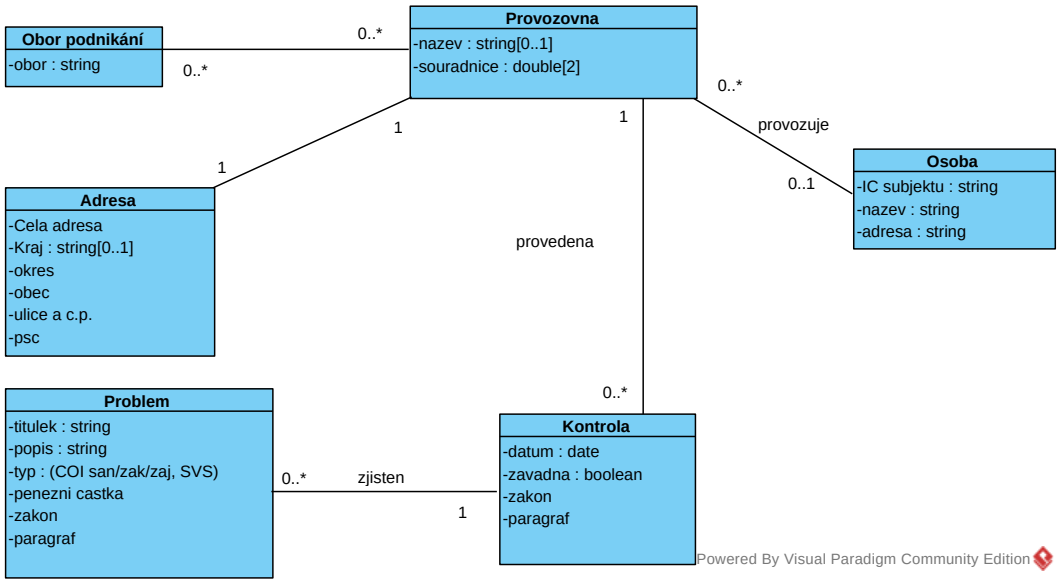
1.3.4 Neúplnost a chybovost dat

Data SVS ani ČOI nejsou zdaleka taková, jaká by si programátor představoval. Z různých právních a jiných důvodů může libovolná část dat z modelu chybět. To platí hlavně pro ČOI. Chybět může například IČO, název, adresa podnikatele, informace o provozovně a další. Navíc se také stává, že data nabývají nepovolených hodnot. Častým příkladem je, když atribut ulice například nabývá hodnoty *"Komenského 12, 77600"*, přestože pro číslo popisné i PSČ existují příslušné atributy a ulice má tedy mít hodnotu pouze *"Komenského"*. Řešením takových anomálií a čištěním dat jsme se nezabývali. Nebylo to pro potřeby naší webové aplikace nutné a ve většině případů by to ani nijak řešit nešlo. Je ovšem dobré o tom vědět.

1.4 Požadavky na integraci

Konceptuální model integrované databáze 1.8 byl navržen tak, aby obsahoval všechny důležité atributy společné pro ČOI a SVS. Cílem bylo zachovat co nejvíce informací a zároveň se zbavit dat, která byla specifická pouze pro ČOI nebo pouze pro SVS a nebyla přínosem pro sjednocená data. Byla to tedy podmnožina sjednocení modelů dat ČOI a SVS, která obsahovala jejich průnik a ještě něco navíc. Konkrétně atributy *zákon, paragraf a peněžní částka* třídy *problém* se nacházejí pouze v datech ČOI, ale do integrované databáze byly zařazené. Považovali jsme je za moc důležité na to, aby byly vynechané. Popisují zajímavé informace a mohly by přijít vhod v budoucnu při rozšiřování integrované databáze o další zdroje. To samé platí pro atributy *zákon a paragraf* třídy *kontrola*. Naopak souřadnice provozovny, název majitele a adresa majitele sice nejsou součástí dat ČOI, ale šlo je získat díky jejich propojení s otevřenými daty získanými pomocí Google geocoding API (viz Google, 2016a) a datovou sadou ARES (Administrativní registr ekonomických subjektů) na doméně Opendata.cz. Zde je přesně vidět využití principu Linked Data. Atributy *titulek a popis* třídy *problém* lze uměle vytvořit z atributů kontrol, zajištění a zákazů ČOI.

Co se týká atributů exkluzivních pro SVS, jedná se za prvé o obor podnikání, který byl vybrán do integrované databáze kvůli své důležitosti. Druhý a poslední je atribut *celá adresa* třídy *provozovna*, který v ČOI není k dispozici a není ani potřeba, protože ČOI poskytuje strukturované informace o adrese. Tento atribut se vyskytuje v některých registračních tabulkách SVS, kde chybí strukturovaná



Obrázek 1.8: Konceptuální model integrované databáze

Třída *Kontrola* představuje kontrolu ČOI nebo SVS. Kontroluje se právě jedna *Provozovna*. Na druhou stranu kontrol jedné provozovny může proběhnout libovolně mnoho. Provozovna má právě jednu adresu, maximálně jednoho vlastníka (*Osoba*) a libovolně mnoho uvedených oborů podnikání. Každá kontrola má uvedených libovolně mnoho *Problémů* a každý problém patří k právě jedné kontrole. Co se týče atributů (*titulek, popis, zákon, peněžní částka, název provozovny, kraj, ulice, psč, datum, název osoby*), prakticky jakýkoliv může chybět (na diagramu to pro lepší přehlednost není explicitně vyznačené).

adresa a proto je součástí modelu.

Atribut *závadná* třídy *provozovna* se nevyskytuje ani v datech ČOI ani SVS, ale je vhodný pro lepší práci s daty a jednoduše se vytvoří. Všechny ostatní atributy byly přímo převzaté z konceptuálních modelů ČOI a SVS.

1.4.1 Očekávané množství dat

Co se týká objemu dat v integrované databázi, nedal se přesně odhadnout jednak kvůli struktuře webových stránek SVS a za druhé kvůli tomu, že při integraci se někteří podnikatelé a provozovny z dat ČOI a SVS sjednotí.

Problém u dat SVS vyplýval z toho, že (před vytěžením) nebylo možné zjistit, které kontroly mají uvedeného podnikatele a IČ nebo naopak, které registrované provozovny a podnikatelé byli kontrolovaní. V nejhorším případě by tedy SVS do integrované databáze nepřispělo žádnými daty. Na základě (rovnoměrně) náhodně vybraných vzorků jsme přibližně odhadli objem dat, kterým by SVS mělo přispět. Tabulka všech registrací SVS obsahuje 15336 řádků. Každý řádek představuje provozovnu, ale některé se opakují. Na základě 160-ti vzorků jsme odhadli, že duplicity tvoří přibližně 20% řádků. Když ještě uvážíme, že ne všechny provozovny byly kontrolované (hrubým odhadem mohlo být kontrolovaných 80% všech registrovaných provozoven), můžeme očekávat, že SVS přispěje 9800 provozovnami. Součet všech počtů kontrol uvedených ve webové aplikaci SVS je 17733. Když zase uvážíme, že ne všechny kontrolované provozovny budeme stahovat, protože nebudou mít uvedené IČ (hrubým odhadem by jich mohlo být 20%), můžeme očekávat, že SVS přispěje 14200 kontrolami (do integrované databáze se budou brát jen kontroly stažených provozoven). Součet všech počtů závadných kontrol činí 753. Na základě 17-ti vzorků jsme odhadli, že je při každé závadné kontrole byla zjištěna cca jedna a půl závady. Proto můžeme očekávat, že SVS přispěje do integrované databáze 1130 závadami/problémy. Horní odhady počtů dat SVS jsou na základě výše zmíněných čísel 15336 provozoven a osob, 17733 kontrol a 3800 závad.

Z dat ČOI jsme zjistili (pomocí SPARQL dotazů), že by měla do integrované databáze přispět 43022 kontrolami, 75434 problémy, 15231 osobami a 57784 peněžními částkami. Další podrobnosti viz testování úplnosti integrace.

Integrací se některé provozovny a podnikatelé sjednotí. Dopředu však nelze vědět, kolik jich bude. Jisté je jen to, že počet podnikatelů by se měl pohybovat mezi 15231 a 30567. Počet provozoven by měl být maximálně 58388, což je součet počtu kontrol ČOI a horního odhadu počtu provozoven SVS. Když budeme předpokládat, že každá provozovna kontrolovaná ČOI byla v datech kontrolovaná průměrně dvakrát, můžeme v integrované databázi očekávat přibližně 30000 provozoven.

1.5 Požadavky na vytěžování

Data ČOI jsou k dispozici jako otevřená data na doméně Opendata.cz. Pro jejich získání nebylo potřeba je nijak vytěžovat nebo měnit. Stačilo je stáhnout a s nimi stáhnout i propojená data ze sady ARES (ohledně podnikatelů) a data se souřadnicemi kontrol, které k datům ČOI původně nepatřila.

Data SVS se publikují ve formě tabulek v HTML a také webové aplikace v HTML s použitím JSONu prostřednictvím AJAJ (AJAJ je to samé jako AJAX, jen místo XML používá JSON. AJAX je technologie asynchronní komunikace webového prohlížeče s HTTP serverem). Data bylo potřeba správně stáhnout, parsovat a převést do vhodné podoby (podrobněji v kapitole *Návrh*).

Nakonec bylo také potřeba určit frekvenci vytěžování. ČOI svá publikovaná data aktualizuje jednou za tři měsíce, zatímco SVS tvrdí, že aktualizace probíhají vždy do 24 pracovních hodin od uskutečnění kontroly. Na základě toho se rozhodlo, že kvůli jednotnosti integrované databáze by bylo vhodné aktualizovat data jednou za tři měsíce hned po aktualizaci dat ČOI na její doméně. Toto rozhodnutí bylo vhodné i proto, že předpokládaná doba vytěžování se pohybovala v desítkách hodin. Bylo by tedy komplikované data aktualizovat často.

1.6 Požadavky na webovou aplikaci

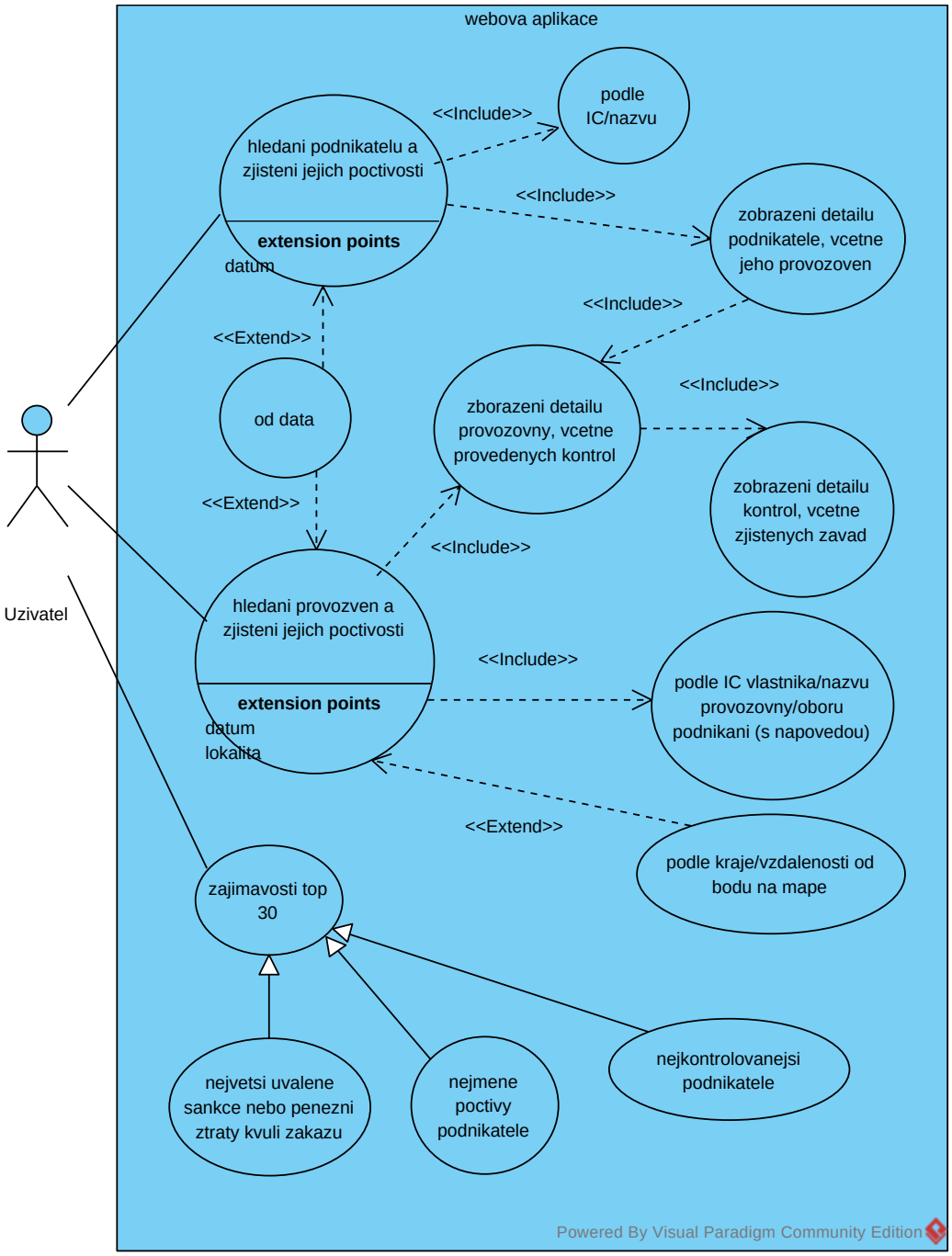
Na základě modelu integrované databáze jsme navrhli požadavky na webovou aplikaci.

1.6.1 Funkční požadavky

Po analýze požadavků integrace 1.4, jsme zvážili možnosti webové aplikace nad integrovanou databází a navrhli funkční požadavky 1.9. Aplikace měla umožnit vyhledávání podnikatelů několika různými způsoby a zjistit jejich poctivost. Mělo se rozlišovat mezi vyhledáváním podnikatelů a vyhledáváním provozoven. Vyhledávání mělo také využívat mapu a hledání pomocí souřadnic. Nakonec mělo být možné zjistit detailnější informace o jednotlivých podnikatelích, provozovnách, kontrolách a závadách.

1.6.2 Kvalitativní požadavky

Z pohledu uživatele jsme zavedli požadavek na přístupnost. Aplikace měla být responzivní a fungovat správně na všech dnes běžných zařízeních. Měla být použitelná jak na počítačích a tabletech, tak i na mobilních zařízeních. Z pohledu programátora byl navíc zaveden požadavek na jednoduché rozšíření integrované databáze přidáním nového datového zdroje. Tím by se rozšířilo pole působnosti webové aplikace.



Obrázek 1.9: Diagram funkcionality webové aplikace

Uživatel bude mít možnost hledat podnikatele nebo provozovny a zjistit jejich poctivost. Dále bude mít možnost zobrazit seznam třiceti nejvyšších sankcí a ztrát, nejméně poctivých a nejkontrolovanějších podnikatelů. Bude možné hledat podnikatele podle IČ nebo názvu firmy a zobrazit seznam jejich provozoven. Provozovny bude možno hledat podle názvu, IČ vlastníka a oboru podnikání. Hledání půjde omezit na provozovny pouze v konkrétním kraji nebo v dané vzdálenosti od bodu na mapě a konkrétního data. U vyhledaných provozoven bude možné zobrazit jednotlivé kontroly a u nich všechny zjištěné závady.

2. Návrh

Po analýze požadavků na integraci, vytěžování dat a funkcionalitu webové aplikace bylo potřeba navrhnout cílovou podobu vytěžovaných a integrovaných dat. Dále musel být navržen způsob vytěžování a převodů dat. Nakonec bylo také nutné navrhnout architekturu a GUI webové aplikace.

2.1 Cílová podoba dat

Vytěžovaná data měla být převáděna do podoby Linked Data. To znamená, že jejich výsledný formát měl být RDF a v rámci RDF měly být strukturované pomocí vhodné kombinace globálně používaných známých slovníků.

2.1.1 Cílová podoba dat ČOI

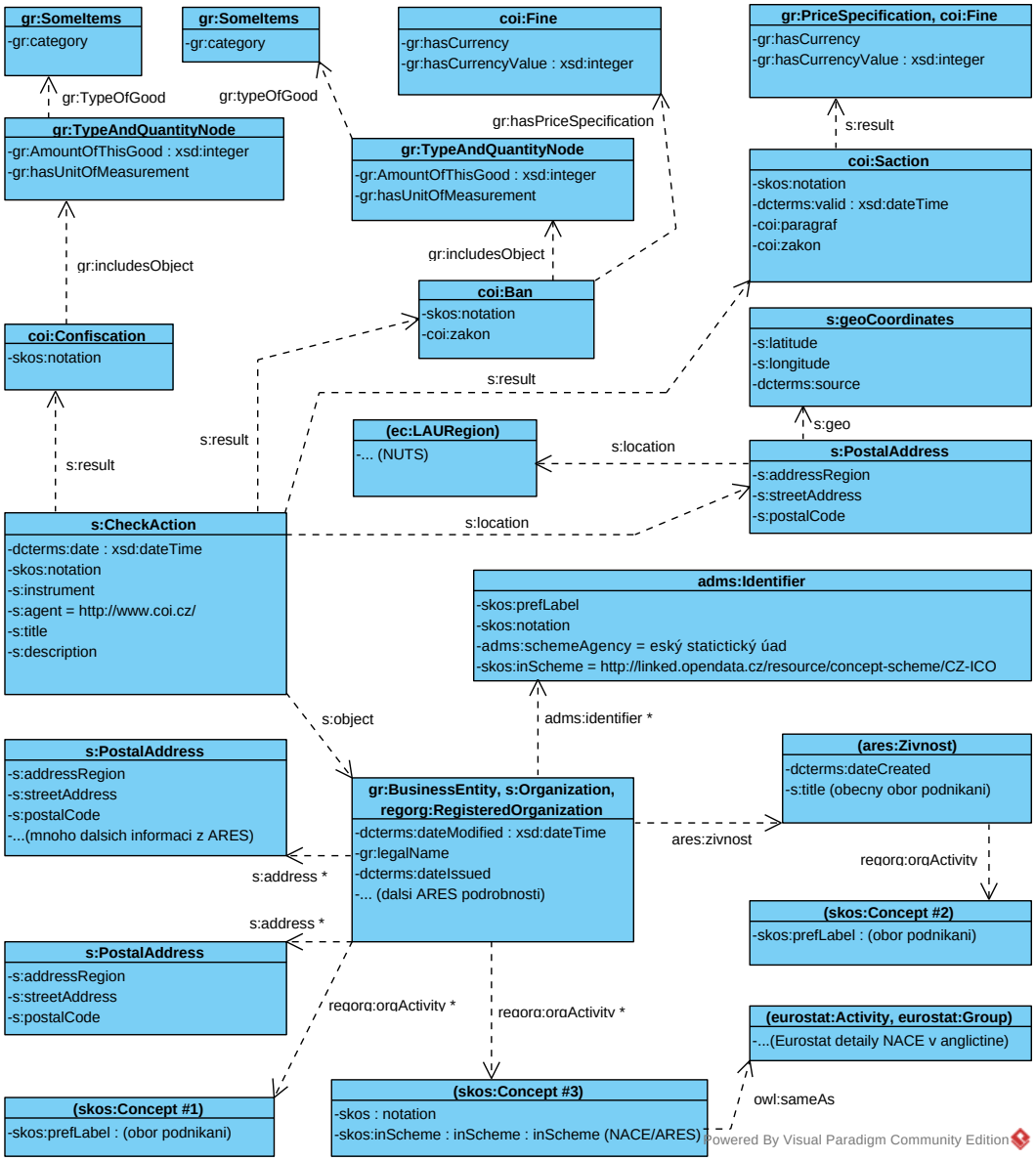
Data ČOI a jejich obohacující data z domény Opendata.cz tuto podobu již měla. Stačilo je zmapovat a metodou reverzního inženýrství získat jejich logický model 2.1.

2.1.2 Cílová podoba dat SVS

Pro data SVS vytěžovaná z jejich webové aplikace bylo potřeba navrhnout vhodnou RDF ontologii. Ontologii SVS 2.2 jsme navrhli tak, aby měla co nejvíce společného s existující ontologií ČOI. Kde nebylo možné inspirovat se ČOI, použili jsme co nejvýstižnější a zároveň co nejznámější slovníky. Jak moc dobře je slovník známý, jsme posuzovali pomocí vhodné aplikace (viz Open Knowledge Foundation, 2016). Pomocí tohoto webu lze vyhledávat známé RDF typy a vlastnosti podle názvu a přibližně zjistit, jak často jsou na internetu používané. Níže je seznam námi užitých známých slovníků.

s: http://schema.org/
xsd: http://www.w3.org/2001/XMLSchema#
gr: http://purl.org/goodrelations/v1#
adms: http://www.w3.org/ns/adms#
dcterms: http://purl.org/dc/terms/
skos: http://www.w3.org/2004/02/skos/core#

Schema.org obsahuje mnoho tříd a vlastností vhodných pro naše data. Jedná se hlavně o popis organizací, adres a kontrol. K tomu slouží vlastnosti a třídy *s:Organization, s:LocalBusiness, s:parentOrganization, s:GovernmentOrganization, s:businessFunction, s:name, s:PostalAddress, s:Place, s:location, s:streetAddress, s:address, s:addressRegion, s:postalCode, s:GeoCoordinates, s:geo, s:latitude, s:longitude, s:legalName, s:CheckAction, s:object, s:result, s:description, s:AuthorizeAction, s:recipient, s:purpose, s:agent.* Jednotlivé vlastnosti mají vysoké hodnocení a jejich kombinace a propojení se přesně hodí k popisu většiny dat SVS. Ostatní podobně známé slovníky, jejichž vlastnosti a třídy by šly použít jako alternativa k výše zmíněným, obsahují vždy jen malou podmnožinu těchto vlastností. Z toho plyne, že primárně nejsou určené k popisu adres, kontrol, organizací atd., ale jen náhodou sdílejí pár vlastností se *Schema.org*. Proto bylo



Obrázek 2.1: Ontologie (logický model) dat ČOI, obohacený převážně o informace z ARES, tak jak jsou publikovaná na doméně Opendata.cz

Hvězdičky zdůrazňují, že takových hran/vlastností z jednoho subjektu může být několik, i když bychom očekávali jen jednu. Obecně musíme počítat s tím, že jakákoliv hrana může chybět nebo naopak může být multiplikovaná. Očíslování některých tříd (např *s:Concept #1*) jen zdůrazňuje, že se nejedná o stejné datové třídy i když jejich RDF typ je stejný. Datové třídy a atributy v závorkách jsou propojená data z domény Opendata.cz, která nepatří do ČOI. Je dobré o nich vědět do budoucna, ale nebudou patřit ani do cílové podoby dat ČOI.

Co se týče konceptuálního významu 1.6 *s:CheckAction* představuje kontrolu, *coi:Confiscation/Ban/Sanction* jsou konfiskace, zákazy a sankce *s:Organization* je osoba s adresou *s:PostalAddress* a IČ *adms:Identifier*. Adresa provozovny má uvedené souřadnice *s:GeoCoordinates* a také kraj *s:addressRegion*, ulici a PSČ.

vhodnější použít *Schema.org*, nehledě na to, že se jedná o velmi známý a dobře udržovaný slovník.

Xsd je univerzálně používaný pro serializaci datových typů (je to de facto standard pro tento účel) a k tomu jsme ho také použili. Mezi typy, které popisuje patří *xsd:integer, xsd:float, xsd:string, xsd:date, xsd:dateTime* atd.

Gr, Adms, Dcterms a Skos jsou použité v ontologii dat ČOI pro popis názvů, IČ a určení času. Stejným způsobem jsme je použili i pro SVS, abychom zajistili co největší podobnost obou modelů.

Nakonec jsme pro velmi specifické atributy a třídy, které nešlo najít v žádném rozumném slovníku, zavedli vlastní, úplně nové vlastnosti a typy.

<http://svs/ontology#obec>
<http://svs/ontology#RegistracniCislo>
<http://svs/ontology#Zavada>
<http://svs/ontology#Chov>
<http://svs/ontology#Provoz>
<http://svs/ontology#ProvozniZavada>
<http://svs/ontology#KomoditniZavada>

Ontologie dat SVS byla navržená tak, aby pro uživatele dat bylo co nejjednodušší pochopit jejich sémantiku. To zajistilo použití kombinace co nejznámějších RDF slovníků v čele se Schema.org. Díky tomu bylo možné data publikovat jako otevřená data ve formátu Linked Data a propojit je s ostatními daty na doméně Opendata.cz, podobně jak jsou propojená data ČOI, NKÚ, ARES a další.

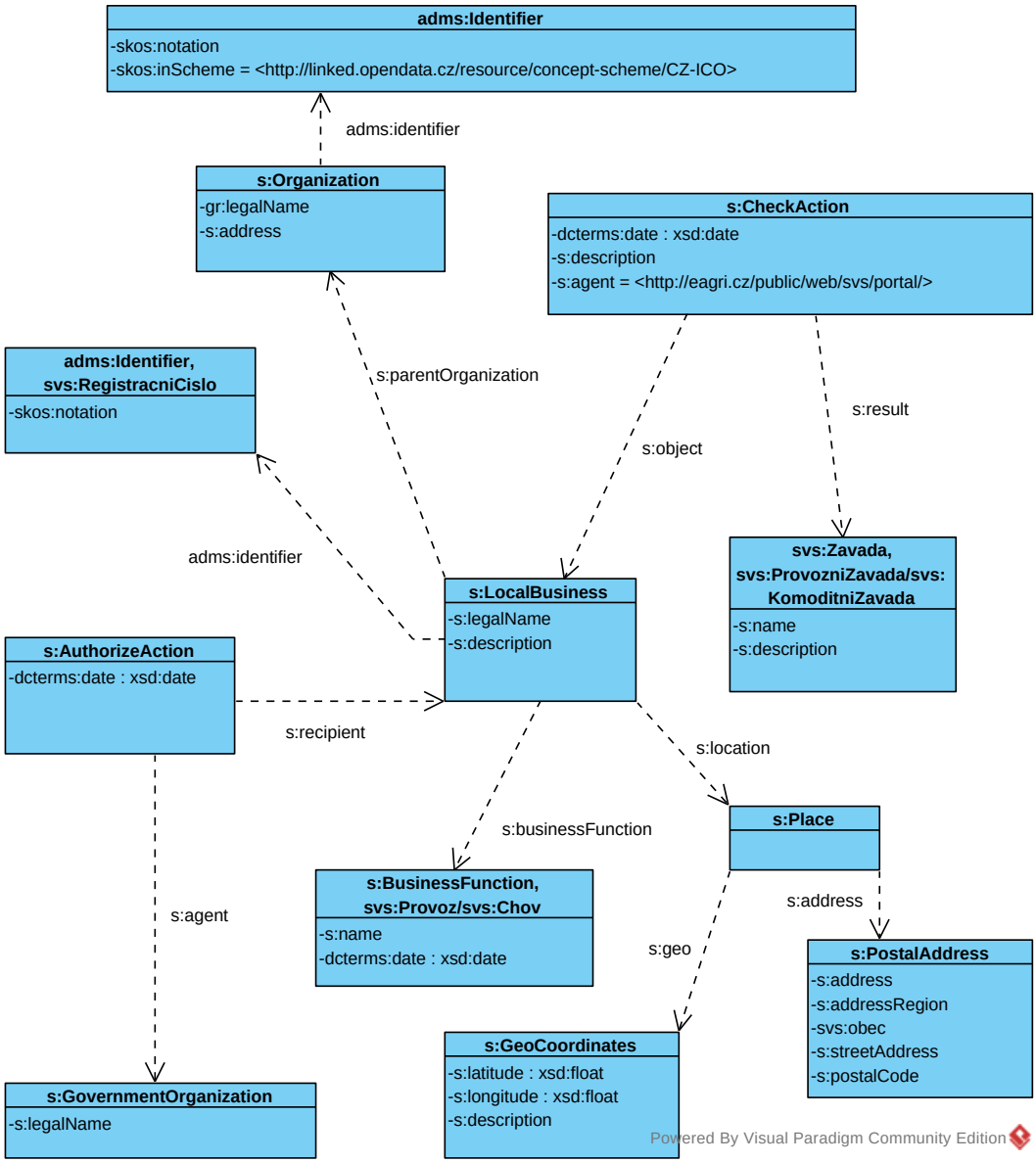
2.1.3 Cílová podoba dat v integrované databázi (APP)

Integrovaná databáze měla sloužit jako zdroj dat webové aplikaci. Proto se o jejím datovém modelu často níže mluví jako o *aplikačním modelu (APP).*

Nejdříve bylo potřeba rozhodnout, jaký logický formát by měla integrovaná databáze mít. Zda by měl být grafový, relační nebo jiný. Nejjednodušší by pravděpodobně bylo, aby formát byl grafový, protože cílové podoby dat ČOI a SVS měly být grafové dle zadání. Z logického hlediska jsou to grafová data ve formátu RDF. Na druhou stranu se už muselo dbát i na požadavky navrhované funkcionality webové aplikace. Víme, že existují servery grafových dat, které lze použít jako backend webové aplikace. Bylo ale potřeba zjistit, zda takové servery podporují práci s prostorovými daty. Ukázalo se, že open-source verze grafového databázového serveru Virtuoso tuto funkcionalitu podporuje. Navíc instance tohoto serveru je na KSI MFF UK dostupná pro využití studenty. Pak už nám nic nebránilo ve volbě grafového formátu jako cílového pro naši integrovanou databázi. Všechnu funkcionalitu webové aplikace by mělo být možné zajistit s použitím grafového formátu RDF.

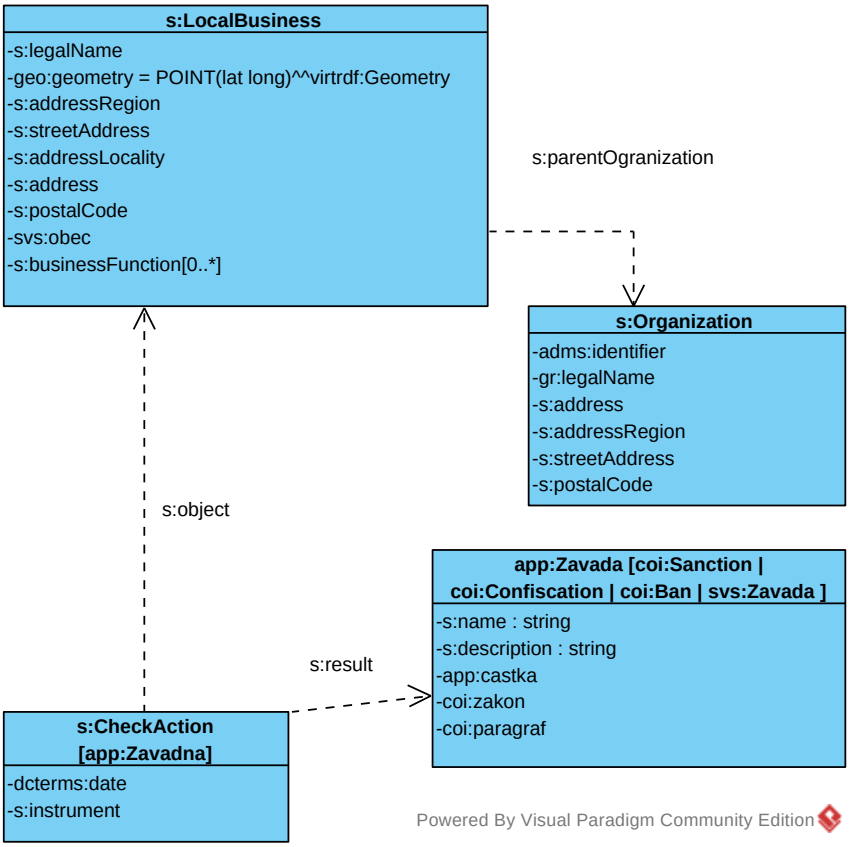
Model dat integrované databáze 2.3 vznikl sjednocením zjednodušených cílových modelů ČOI a SVS. APP model byl navržen, aby byl co nejjednodušší. Zároveň ovšem musel být dost strukturovaný na to, aby pomocí něj šla naimplementovat veškerá navrhovaná funkcionalita. Jedna z výhod vytvoření tohoto modelu byla, že umožnil nezávislý vývoj webové aplikace a implementace vytěžování dat.

Na rozdíl od cílového modelu dat SVS nebyl model APP navržen pro veřejnost, ale spíše pro interní potřeby naší webové aplikace.



Obrázek 2.2: Navržená ontologie SVS, ve které chceme data vytěžovat

Co se týče konceptuálního významu 1.5 *s:CheckAction* představuje kontrolu s uvedenými závadami *svs:Zavada*. Závada má titulek *s:name* a slovní popis *s:description*. *s:LocalBusiness* je provozovna a patří podnikateli *s:Organization* s IČ *adms:Identifier*. Provozovnu identifikuje registrační číslo *adms:Identifier*, *svs:RegistracniCislo* a její provoz schvaluje autorita *s:GovernmentOrganization* pomocí tzv. autorizační akce *s:AuthorizeAction*. Obor podnikání provozovny uvádí *s:BusinessFunction* a její lokaci *s:Place*, *s:GeoCoordinates*, *s:PostalAddress*.



Obrázek 2.3: Logický model integrované databáze

Co se týče konceptuálního významu 1.8 *s:CheckAction* představuje kontrolu, která může mít uvedené problémy/závady *app:Zavada*. Kontrolují se provozovny *s:LocalBusiness*, které patří podnikatelům/osobám *s:Organization*. Podnikatelé mají uvedené IČ *adms:identifier*, název *gr:legalName* a adresu. Provozovny mají uvedený název *s:legalName*, souřadnice *geo:geometry*, obor podnikání *s:businessFunciton* a další.
Až na pár výjimek lze většinu atributů získat přímo (nebo po potřebné transformaci) z dat ČOI i SVS. To neplatí pro zákony, které překročila nějaká závada *coi:zakon, coi:paragraf* a sankce nebo ztráty kvůli zákazu *app:castka*, které lze získat pouze z dat ČOI.

2.2 Návrh vytěžování a převodů dat

Nezávisle na technickém způsobu vytěžování bylo potřeba přesně zvolit, která data by se měla vytěžovat a navrhnout jak. K tomu bylo také potřeba zjistit, kde se data nachází - na jakých serverech, v jakých souborech a v jakém formátu.

2.2.1 Obecné řešení problému spojování dat

Při sjednocování množin podobných dat v RDF z mnoha nezávislých zdrojů je nutné data správně integrovat. Například pokud by se do výsledné databáze dostaly informace o stejném podnikateli z více zdrojů, je potřeba vyjádřit, že se týkají jednoho a toho samého podnikatele a nikoliv dvou různých. Tuto situaci jsme v RDF navrhli řešit použitím stejného URI. Například pro osoby by se URI generovalo na základě jejich IČ (vždy stejných způsobem). Tento přístup zajistí správné spojení dat ve výsledné databázi. Je ovšem potřeba si uvědomit, že tak mohou vzniknout duplicity.

2.2.2 Návrh tranformací dat ČOI

Data ČOI nebylo potřeba nijak měnit ani transformovat, jednoduše je stačilo stáhnout tak, jak byla na doméně Opendata.cz ve formátu RDF. K tomu se musela zjistit URI RDF grafů, ve kterých byla data ČOI uložená (z publikačního systému Opendata.cz). Pak bylo nutné zjistit URI grafů obohacujících dat (ARES a souřadnice) a stáhnout z nich jen ty RDF data, která nás zajímala (pomocí dotazů SPARQL). Datová sada ARES totiž obsahuje mnoho dalších informací, které se naší práce netýkají. Níže je seznam grafů (URI), ve kterých se data nacházejí.

- ČOI kontroly - Kontroly České obchodní inspekce
  http://linked.opendata.cz/resource/dataset/coi.cz/kontroly
- ČOI zákazy - Zákazy České obchodní inspekce
  http://linked.opendata.cz/resource/dataset/coi.cz/zakazy
- ČOI zajištění - Zajištění České obchodní inspekce
  http://linked.opendata.cz/resource/dataset/coi.cz/zajisteni
- ČOI sankce - Sankce České obchodní inspekce
  http://linked.opendata.cz/resource/dataset/coi.cz/sankce
- Zaměření kontrol České obchodní inspekce
  http://linked.opendata.cz/resource/dataset/coi.cz/zamereni
- ARES - Datový soubor s daty Identifikačních čísel (IČ)
  http://linked.opendata.cz/resource/dataset/ic
- Google Geocoding souřadnice kontrol ČOI *(Toto není oficiální název, protože tento graf není oficiálně veřejně dostupný na doméně Opendata.cz. Při zjišťování jeho URI bylo potřeba trochu vynalézavosti.)*
  http://linked.opendata.cz/resource/dataset/coi.cz/geocoordinates/google

Poslední dva grafy představují obohacující informace k datům o kontrolách ČOI. Z grafu ARES bychom stáhli pouze název podnikatele, jeho adresu a IČ. Z grafu se souřadnicemi bychom stáhli vše.

2.2.3 Návrh tranformací dat SVS

Data SVS se dělí na dvě části. Jedna jsou registrace provozoven a druhá informace o proběhlých kontrolách a závadách. Obě jsou dostupné na webu SVS, ovšem nikoliv jako jedna hezká velká tabulka, ale jako šest tabulek s registracemi (viz Státní veterinární správa, 2016c) a jedna dynamická webová aplikace (viz Státní veterinární správa, 2016b). V obou případech je potřeba ještě sledovat jeden nebo dva odkazy, abychom se dostali ke všem informacím, které lze ze stránek vyčíst. Ne všechny nás ovšem zajímaly - vybrali jsme pouze šest tabulek z asi třiceti a to jen ty, které obsahovaly IČ podnikatelů. Ostatní jsme ignorovali, protože by jich jednak bylo neúnosně moc a také by bylo nemožné je spojit s daty ČOI kvůli absenci IČ. Níže je seznam tabulek vybraných pro extrakci a zkratek, které jsme zavedli pro jejich dlouhé názvy.

- EU 1 - Zpracovatelé živočišných produktů schválení a registrovaní pro obchodování v rámci EU
- EU 2 - Zpracovatelé živočišných produktů dočasně schválení a registrovaní pro obchodování v rámci EU
- EU 4 - Schválené produkční podniky akvakultury
- EU 10 - Seznam mléčnic v působnosti inspektorátu
- OO 5 - Producenti neharmonizovaných komodit
- PP 1 - Zpracovatelé živočišných produktů registrovaní pro přímý prodej v ČR

Pro každou registrační tabulku a také pro webovou aplikaci jsme navrhli způsob, jak data vytěžovat.

Co se týče tabulek, způsob extrakce jejich dat byl skoro u každé dost podobný. Přesto se vždy našel nějaký nový problém, který extrakci dat z dané tabulky odlišoval od ostatních. Obecně ale bylo vždy potřeba stáhnout hlavní HTML tabulku, která obsahovala registrační číslo, název, drobněji strukturovanou adresu, případně typ provozu nebo druh chované zvěře, datum schválení a také odkaz na detailnější informace. Pro každý řádek bylo potřeba stáhnout data z ještě jedné malé HTML tabulky, přístupné přes zmíněný odkaz, která obsahovala detailnější popis provozovny. Odtud bylo potřeba vyparsovat především název majitele, jeho IČ, případně adresu, datum schválení provozovny a název schvalující autority. Všechny zmíněné HTML tabulky byly dostupné pomocí HTTP dotazů GET a data šla parsovat s použitím CSS selektorů.

V neposlední řadě bylo potřeba zajistit správné spojování dat o podnikatelích. Jednoduše se mohlo stát, že jedna osoba vlastnila dvě mléčnice a zároveň jeden rybník. V tomto případě bychom zvlášť stáhli seznam mléčnic a zvlášť seznam rybníků, tím by vznikly dva záznamy o té samé osobě a ty bylo potřeba spojit. Bylo také nutné uvědomit si, že mohou vzniknout duplicity spojováním dat.

Například pokud by jeden podnikatel vlastnil jednu mlékárnu a jeden rybník, je velmi pravděpodobné, že ve výsledných datech by měl dvakrát zmíněné jméno, adresu atd. Přitom hodnoty by se teoreticky mohly lišit, pokud by data v jedné z tabulek byla zastaralá nebo chybná.

Co se týče konkrétních tabulek, u *EU 1* bychom nestahovali číslo pro úřední evidenci, protože nevíme, co znamená a pouze by znepřehledňovalo data. Další tabulka *EU 4* je strukturovaná jinak než ostatní. Její řádky obsahují informace o hlavní provozovně, pod kterou spadá několik rybníků. Každý rybník ovšem také představuje jednu provozovnu. Od hlavní provozovny vede odkaz na její rybníky. Bylo potřeba stáhnout data o hlavní provozovně, pro každou z nich pak následovat odkazy a stáhnout data o jejich rybnících. Stahovali bychom vše, až na zeměpisnou polohu rybníků a informace o nákazách a zamořeních. Zeměpisnou polohu bychom získali z webové aplikace, takže by bylo zbytečné ji stahovat dvakrát. Informace o nákazách jsou velmi specifické a jen by znepřehledňovaly stahování. Nakonec u tabulky *EU 10* bychom nestahovali kódy hospodářství, protože nevíme, co znamenají a pouze by znepřehledňovaly data.

Původní návrh vytěžování dat z webové aplikace (MAPA)

Jelikož webová aplikace SVS zobrazuje výsledky kontrol na mapě, budeme jí někdy zkráceně označovat jako *MAPA*.

Vytěžování dat o kontrolách z webové aplikace SVS by požadovalo poněkud odlišné kroky od vytěžování dat z registračních tabulek. Bylo by potřeba nejdříve stáhnout seznam map 2.5. Ke každé mapě by se pak stáhl seznam evidovaných období 2.6. Oba seznamy jsou dostupné ve formátu JSON a webová aplikace je zobrazuje pomocí technologie AJAJ (jak bylo zjištěno analýzou webové aplikace). Pro každý úsek, kde úsekem se myslí dvojice (mapa, období), by pak bylo potřeba stáhnout seznam bodů v daném úseku. Bodem se myslí provozovna, její souřadnice a odkaz na její kontroly v daném úseku. Tato terminologie vznikla po analýze dat webové aplikace, kde se právě toto označení používá. Seznam bodů v nějakém úseku je také dostupný ve formátu JSON. Nakonec by zbývalo pro každý jednotlivý bod stáhnout detailní informace o závadách 2.4, které v daném bodě proběhly v rámci konkrétního úseku. Informace o závadách obsahují datum, typ závady (komoditní nebo provozní), krátké shrnutí a nezkrácený popis. Za předpokladu, že v jednom dni proběhla maximálně jedna kontrola, by bylo možné takto vytěžit všechny informace o kontrolách, závadách a souřadnice provozoven.

V neposlední řadě by bylo potřeba na základě počtu nezávadných kontrol vytvořit jednotlivé uzly reprezentující nezávadné kontroly (pozn.: v aplikaci SVS se místo kontrola používá pojem akce a závadným kontrolám se říká pozitivní akce). SVS totiž nerozlišuje jednotlivé kontroly a u nezávadných kontrol uvádí pouze jejich počet. Navíc by se k nezávadným kontrolám ještě mělo přidat nepřesné datum odvozené od daného úseku. Proběhla-li nezávadná kontrola někdy v březnu roku 2016, je možné ji alespoň přibližně časově zařadit a přidat k ní datum 1.3.2016.

JSON data jsou dostupná pomocí HTTP dotazů POST. Níže jsou ukázky parametrů potřebných pro stažení zmíněných množin dat.

http://eagri.cz/public/app/svs_pub/mapy_vk/bin/maps_read_data.php

POST type: seznam_map

Obrázek 2.4: Detailní informace o závadách provozovny ve fromátu JSON

POST type: obdobi; mapa: 01
POST type: data_bodu; mapa: 01; obdobi: brezen16
POST type: detail; bod_id: 9335

Nečekané potíže

Velmi hlubokou analýzou webové aplikace ovšem vypluly na povrch problémy, kvůli kterým musel být původní návrh zavržen.

- Při stahování detailních informací o závadách (detailů bodů) se zjistilo, že některá data nejsou ve validním formátu JSON a tudíž je nelze číst. Jako důsledek nebylo možné pak zajistit přesné testování úplnosti.

- Detailních informací o závadách je mnoho a při stahování velkého množství detailů pomocí HTTP dotazů POST by server SVS přestal odpovídat. Velkým množstvím je myšleno přibližně deset tisíc dotazů. Při větším množství dotazů by server vrátil chybový kód 503 protokolu HTTP, oznamující nedostupnost služby a po nějakou dobu by nebylo možné pokračovat v dalším stahování. Z tohoto důvodu by bylo nutné stahovat data webové aplikace SVS po menších kusech a s dostatečnými časovými rozestupy.

- Pro stahování detailních informací o závadách (detailů bodů) je potřeba použít interní identifikátor SVS tzv. *bod_id*, jak již bylo zmíněno. Zjistilo se ovšem, že tento identifikátor se mění. Změny identifikátoru mohou probíhat s frekvencí cca jednou denně a co je zajímavé, mohou se týkat libovolně starých kontrol. Změny identifikátoru byly zjištěné u kontrol starých i několik měsíců. Původ změn je neznámý.

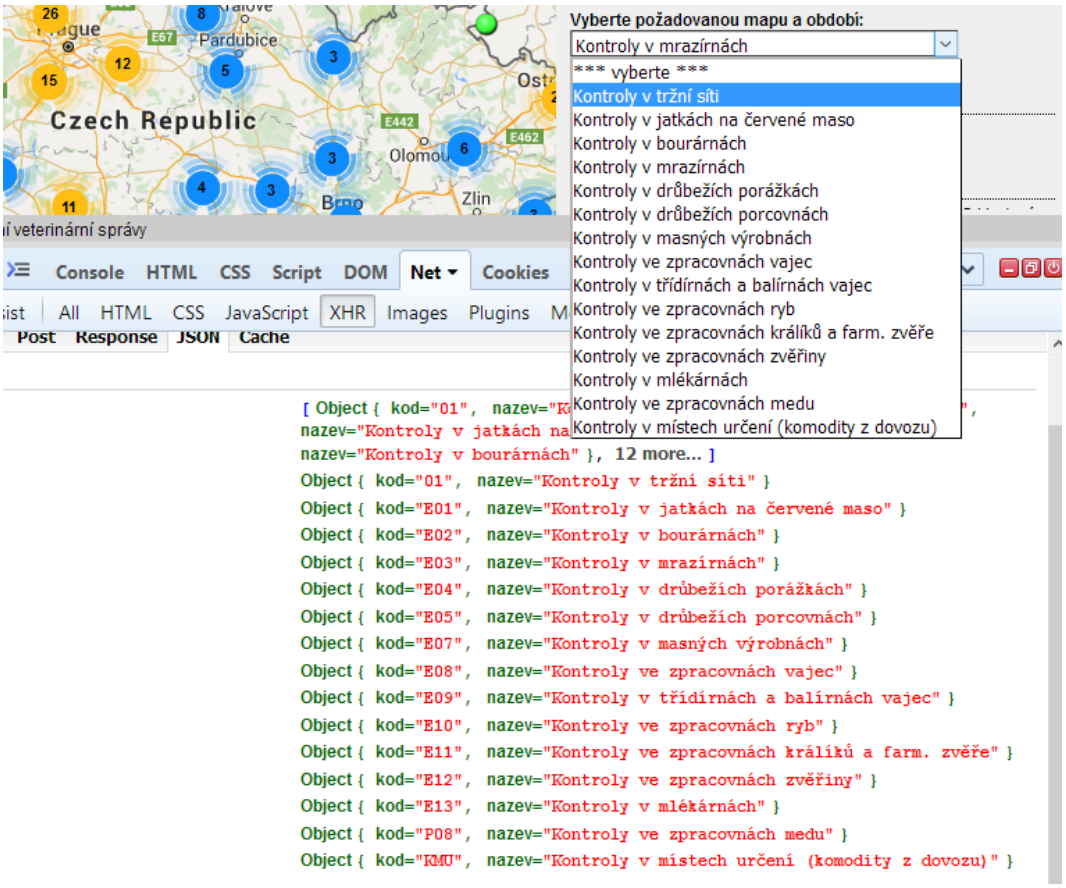
  Za předpokladu, že změny probíhají na základě manipulace z databází SVS, když některý inspektor přidává záznamy o nových kontrolách, by mělo být možné vyhnout se změnám stahováním dat přes víkend nebo přes noc. Tehdy by inspektoři neměli být v práci a neměli by manipulovat s databází. Příklady zaregistrovaných změn identifikátorů u kontrol v tržní síti:

  CZ 42931582 (registracni cislo provozovny)
  17.6.2016 12:17 9718 (datum, cas, bod_id)
  18.7.2016 12:41 9756
  19.7.2016 17:53 9760

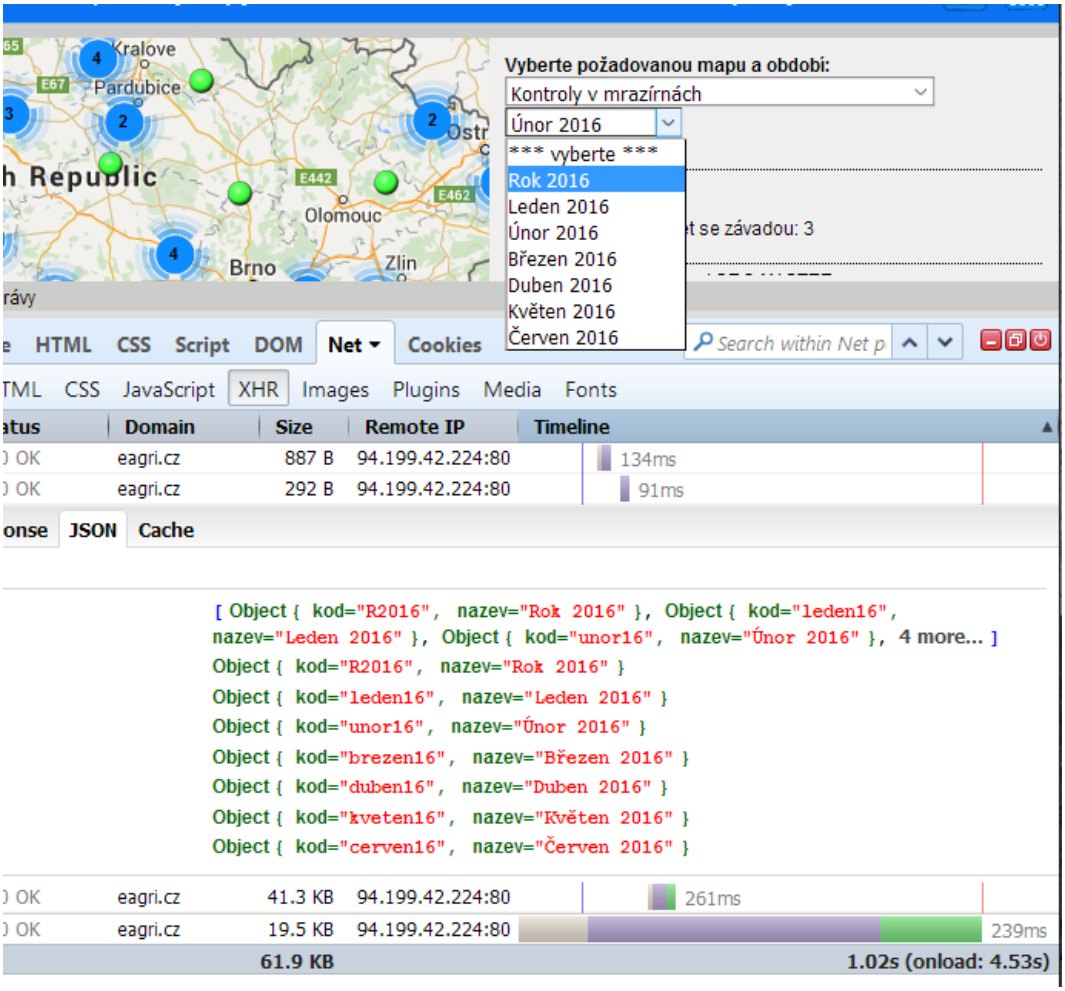
  CZ 21354
  17.6.2016 12:17 9554
  18.7.2016 12:41 9592
  19.7.2016 17:53 9596

Nový návrh vytěžování dat z webové aplikace

Jako reakce na nečekané potíže jsme navrhli nový způsob vytěžování dat SVS z jejich webové aplikace. Místo stahování všech úseků najednou, by se na začátku ručně zvolil 2.5 2.6 jen jeden úsek a ten by byl stažen. Pro stažení všech dat, by se tedy úseky stahovaly postupně každý zvlášť.



Obrázek 2.5: Volba mapy, na základě jejího kódu v JSON souboru z AJAJ



Obrázek 2.6: Volba období, na základě jeho kódu v JSON souboru z AJAJ

Výhody a nevýhody nového návrhu

Nevýhodou nového návrhu je, že by byla potřeba při každém stahování ručně zadávat kód úseku, který se má stáhnout. To ovšem tolik nevadí, protože i u původního návrhu bylo nutné ručně zadávat nepřesné datum, které by se uměle přidalo nezávadným kontrolám. Nezávadné kontroly totiž v datech SVS datum uvedené nemají a obecně jej nelze automaticky odvozovat (šlo by to pouze za přidání určitých předpokladů). Další nevýhodou je, že by bylo potřeba každý požadovaný úsek stáhnout zvlášť, což by bylo časově náročné pro toho, kdo bude vytěžování spouštět.

Na druhou stranu výhodou je, že by se celá transformace zjednodušila a bylo by také možné lépe testovat úplnost. Další výhodou by bylo rychlejší stahování, což by byla jediná možnost jak data stáhnout, kdyby nebyl splněn předpoklad, že se identifikátory bodů o víkendu nemění. Nakonec by také zmizel problém přetěžování serveru SVS a navíc by transformace byla díky své jednoduchosti méně náchylná k dalším nečekaným potížím, které mohou ve webové aplikaci SVS kdykoliv vzniknout.

2.2.4 Návrh tranformací do modelu APP

Pro integraci dat by bylo potřeba zvlášť převést stažená data ČOI a vytěžená data SVS do integrovaného modelu. K tomu jsme navrhli postup ve dvou krocích. Nejdříve by se převedla data ČOI a SVS do modelu APP v jakési dočasné nespojené podobě a posléze by se zajistilo jejich spojení.

Za prvé bylo potřeba spojit osoby z dat ČOI s těmi ze SVS. Za druhé by se měly podobným způsobem spojit provozovny jedné osoby z obou datových sad. Otázka byla, jaké pro ně vybrat společné URI. To bylo problematické, protože SVS sice poskytuje adresu, název a registrační číslo provozovny, ale ČOI naopak žádná registrační čísla nepoužívá a název provozoven nezveřejňuje. Zbývají adresy, ale s těmi je vždy stejný problém, často jsou zadané špatně, často chybí nějaká jejich část, prostě se s nimi pracuje špatně. Řešením bylo použít IČO vlastníka a souřadnice provozovny. Provozovny jedné osoby jsme považovali za stejné, pokud měly stejné souřadnice po první tři desetinná místa (což dává prostor pro chybu až cca 110 m v zeměpisné šířce a až cca 70 m v zeměpisné délce). Ano, je zde problém, pokud někdo vlastní dvě provozovny hned těsně vedle sebe nebo obě ve stejné budově. Na druhou stranu je toto řešení flexibilnější vůči malým odchylkám/chybám v zadaných souřadnicích. Bylo potřeba zvolit nějaký kompromis. Do budoucna je dobré vědět o tom, že slévání by šlo zlepšit zkoumáním adresy, ale není tam žádná garance (z již zmíněných důvodů).

Převod ČOI do APP

Při zkoumání logického modelu dat ČOI bylo důležité si všimnout, že u každé kontroly je uvedená nová adresa. To znamená, že kontrolami jedné provozovny vznikalo mnoho duplikací její adresy. S tím jsme se při tvorbě integrované databáze museli vypořádat. Zdá se, že důvodem byl důraz na informace o podnikatelích, nikoliv o provozovnách. V ČOI si podnikatelé své provozovny nijak neregistrují ani se nezaznamenávají jejich názvy. Navržené řešení bylo, na základě každé lokace kontroly vytvořit virtuální provozovnu. Přičemž několik virtuálních pro-

vozoven by vždy reprezentovalo jednu opravdovou provozovnu. Později, v kroku *spojení dat* by se virtuální provozovny a provozovny z dat SVS správně spojily a tím by vznikly opravdové provozovny.

Kromě vytváření virtuálních provozoven by převod obsahoval také vybírání pouze těch dat z ČOI, která patří do modelu APP. Dále přejmenování příslušných RDF vlastností a typů, kde by to bylo potřeba a nakonec vytvoření popisu závady na základě strukturovaných dat o sankci, zajištění nebo zákazu ČOI.

Převod SVS do APP

Model APP je podobný modelu SVS. Díky tomu by pro převod stačilo jen vybrat ta data ze SVS, která patří do modelu APP, ostatní ignorovat a nakonec přejmenovat několik RDF vlastností a typů. Stejně jako u převodu ČOI do APP by bylo potřeba přestrukturovat souřadnice.

Spojení dat

U dat v integrované databázi bychom se chtěli vyhnout duplicitám, aby webová aplikace s daty mohla pracovat jednoduše a přívětivě pro uživatele. V datech ČOI a SVS ve formátu RDF je ovšem duplikace častá. Například jméno firmy může být uvedeno třikrát, pokaždé trochu jinak (např. "Billa", "Billa s.r.o.", ""). K duplikaci stejných hodnot nedochází, RDF databáze se totiž chová jako množina a neobsahuje unikátní RDF trojici dvakrát. Duplikace je nemožné odstranit automaticky. Nelze na základě nějakého pravidla rozhodnout, která data jsou správnější nebo přesnější. Řešením tohoto problému může být výběr vždy té hodnoty, která je nejdelší co do počtu znaků. Toto řešení samozřejmě není vždy korektní, ale pro potřeby naší integrované databáze (na základě toho, s jakými daty pracujeme) by mělo být ve valné většině případů v pořádku.

Pro spojení dat a zakončení integrace by bylo potřeba spojit dohromady zvlášť osoby a provozovny z dat ČOI a SVS. Posléze by se měly odstranit vzniklé duplicity a také všechny duplicity z původních dat. Nakonec by se ještě k závadným kontrolám přidala informace, že jsou závadné. To by se zajistilo jednoduše na základě počtu závad každé kontroly.

2.2.5 Časové rozvržení převodů

Data ČOI se aktualizují každé tři měsíce, takže je vhodné je v takových intervalech stahovat. Data v registračních tabulkách SVS nemají uvedenou frekvenci aktualizací, ale pravděpodobně se aktualizují kdykoliv se nějaká nová provozovna zaregistruje nebo jiná odregistruje. Je možné je stahovat kdykoliv. Data ve webové aplikaci SVS se aktualizují vždy do 24 pracovních hodin od provedení kontroly. Předpokládáme, že kontroly probíhají jen v pracovním týdnu a jen od 8:00 do cca 16:00. Abychom se vyhnuli výše zmíněným problémům při vytěžování, bylo by vhodné stahovat data pouze přes víkend (případně přes noc v pracovní dny).

Pro zachování časové konzistence integrované databáze jsme zvolili řešení stahovat všechna data jednou za tři měsíce. Přitom u dat SVS by se muselo dbát i na zmíněné časové omezení.

2.3 Návrh webové aplikace

Architekturu webové aplikace jsme navrhli zcela běžných způsobem. Jako zdroj dat měla sloužit databáze s integrovanými daty. V našem případě by se mělo jednat o RDF databázi. Ta byla nejvhodnější, jelikož cílový formát integrovaných dat měl být RDF. Databáze měla mít povolené jen čtení, tím by se vyřešila bezpečnost. Webový server, kde by běžela naše webová aplikace, by se databáze dotazoval a dostával by odpovědi. Webový klient by se zase dotazoval na webový server a jako odpovědi by dostával webové stránky vygenerované na serveru na základě dat z databáze.

2.3.1 Návrh grafického uživatelského rozhraní

Dle požadované funkcionality 1.6.1 jsme připravili náčrt grafického uživatelského rozhraní webové aplikace 2.7 2.8 2.9 2.10 2.11, který souhlasí s touto funkcionalitou. Podle náčrtu se později ubíral vývoj GUI webové aplikace.

2.3.2 Scénáře použití webové aplikace

Na základě návrhu GUI a návrhu funkcionality webové aplikace vzniklo několik scénářů nebo také příkladů použití webové aplikace koncovým uživatelem. Aplikace později byla testovaná tak, aby její funkcionalita při těchto scénářích byla korektní.

- Uživatel zadá hledání osob, zvolí jako kritérium IČ a zadá hodnotu konkrétního IČ 2.7. Jako výsledek se mu zobrazí IČ, název, odkaz na detail a míra poctivosti daného podnikatele nebo oznámení, že daný podnikatel není evidovaný v databázi 2.8.

- Uživatel zadá hledání osob, zvolí jako kritérium IČ a špatně zadá hodnotu konkrétního IČ (nebude to přesně osm číslic). Při pokusu o hledání se uživateli zobrazí upozornění, že zadal špatnou hodnotu, hledání se nevykoná a uživatel dostane příležitost zadat novou, správnou hodnotu.

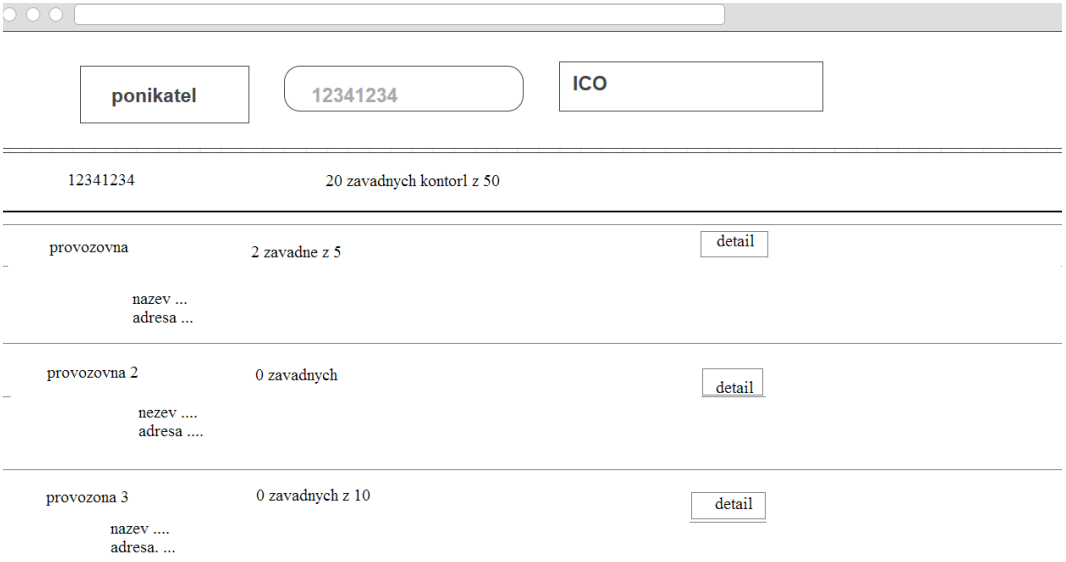
- Uživatel zadá hledání osob, zvolí jako kritérium název a zadá hodnotu konkrétního názvu podnikatele. Jako výsledek se mu zobrazí IČ, název, odkaz na detail a míra poctivosti daného podnikatele nebo oznámení, že daný podnikatel není evidovaný v databázi.

- Uživatel zadá hledání osob, zvolí jako kritérium název a nezadá žádnou hodnotu. Při pokusu o hledání se uživateli zobrazí upozornění, že je potřeba zadat nějaký název, hledání se nevykoná a uživatel dostane příležitost zadat novou, správnou hodnotu.

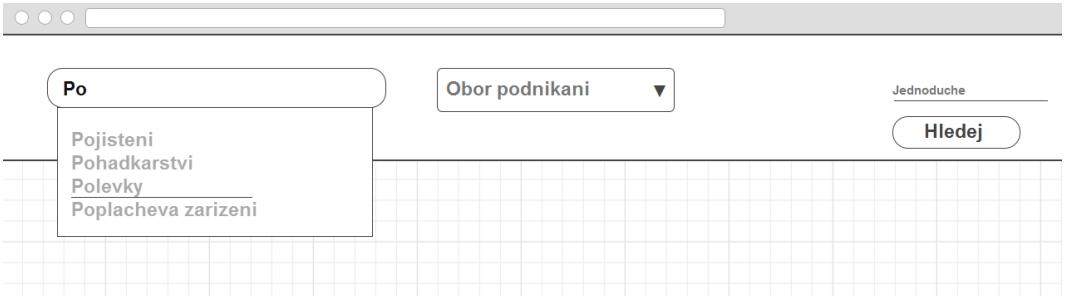
- Uživatel zadá hledání osob a nezvolí žádné další kritérium. Jako výsledek se mu zobrazí IČ, název, odkaz na detail a míra poctivosti všech podnikatelů v integrované databázi.

- Uživatel zadá hledání osob a nezvolí žádné další kritérium. V nabídce rozšířeného vyhledávání uživatel zadá, že nemá zájem o informace starší než

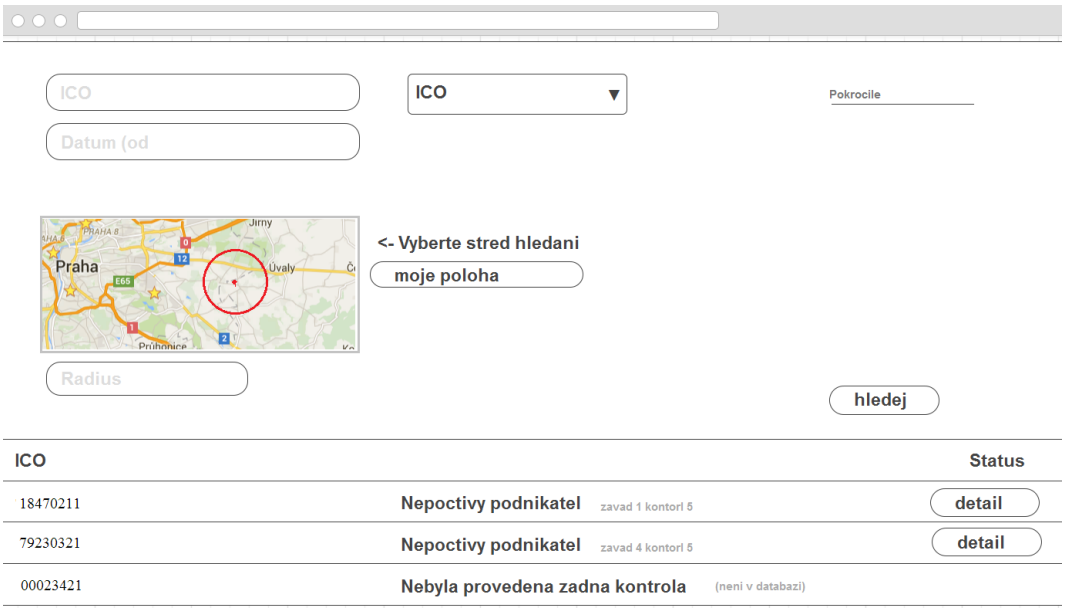
Obrázek 2.7: Hlavní stánka webové aplikace



Obrázek 2.8: Výsledky hledání



Obrázek 2.9: Detail - nápověda u některých políček



Obrázek 2.10: Rozšířené hledání

Obrázek 2.11: Detail dané provozovny, jejích kontrol a jednotlivých závad

je nějaké konkrétní datum. Jako výsledek se mu zobrazí IČ, název, odkaz na detail a míra poctivosti všech podnikatelů v integrované databázi, kde ovšem poctivost bude počítaná pouze na základě kontrol novějších než jaké bylo zadané datum.

- Uživatel zadá hledání provozoven, zvolí jako kritérium obor podnikání a zadá hodnotu konkrétního oboru 2.7. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti dané provozovny nebo oznámení, že daná provozovna není evidovaná v databázi.

- Uživatel zadá hledání provozoven, zvolí jako kritérium název a zadá konkrétní název provozovny. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti dané provozovny nebo oznámení, že daná provozovna není evidovaná v databázi.

- Uživatel zadá hledání provozoven, zvolí jako kritérium název, ale nezadá žádný název. Při pokusu o hledání se uživateli zobrazí upozornění, že by měl zadat nějaký název, hledání se nevykoná a uživatel dostane příležitost zadat novou, správnou hodnotu.

- Uživatel zadá hledání provozoven, zvolí jako kritérium IČ a zadá konkrétní IČ nějakého podnikatele. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti všech provozoven, které daný podnikatel vlastní nebo oznámení, že provozovny tohoto podnikatele nejsou evidované v databázi.

- Uživatel zadá hledání provozoven a nezvolí žádné další kritérium. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti všech provozoven v integrované databázi.

- Uživatel zadá hledání provozoven a nezvolí žádné další kritérium. V nabídce rozšířeného vyhledávání uživatel zadá, že nemá zájem o informace starší než je nějaké konkrétní datum. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti všech provozoven v integrované databázi. Ovšem míra poctivosti bude počítaná pouze na základě kontrol, které proběhly po zadaném datu.

- Uživatel zadá hledání provozoven a nezvolí žádné další kritérium. V nabídce rozšířeného vyhledávání uživatel zadá, že má zájem pouze o provozovny z konkrétního kraje ČR. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti všech provozoven v integrované databázi, které se nacházejí v daném kraji.

- Uživatel zadá hledání provozoven a nezvolí žádné další kritérium. V nabídce rozšířeného vyhledávání uživatel zadá, že má zájem pouze o provozovny v okolí 50 km kolem centra Prahy 2.10. Jako výsledek se mu zobrazí IČ, název provozovny, odkaz na detail a míra poctivosti všech provozoven v integrované databázi, které se nacházejí v dané oblasti.

- Uživatel v navigaci klikne na *top 30* a zobrazí se mu seznamy zajímavostí. Mezi nimi bude třicet nejdražších sankcí a třicet zákazů s nejvyšším finančním dopadem.

- Po vyhledání osoby uživatel klikne na odkaz na detail. Jako výsledek se mu zobrazí informace o osobě včetně seznamu všech provozoven této osoby a odkazů na detaily provozoven.
- Po vyhledání provozovny uživatel klikne na odkaz na detail. Jako výsledek se mu zobrazí informace o dané provozovně včetně její adresy a seznam všech kontrol, které na dané provozovně proběhly. Každá kontrola bude mít uvedené také všechny nalezené závady, pokud nějaké nastaly 2.11.

3. Programátorská dokumentace

Po přípravě návrhů bylo potřeba zvolit technologie pro implementaci datových převodů, integrované databáze a také webové aplikace. Následovala implementace vytěžování dat SVS a stahování dat ČOI. Vhodnými transformacemi pak byla vytvořená integrovaná databáze a nad ní posléze implementovaná webová aplikace. Všechno probíhalo dle návrhů z minulé kapitoly a bylo proti nim testováno.

3.1 Volba způsobu implementace transformací a vytěžování dat

Pro implementaci vytěžování dat jsme zvolili ETL nástroj UnifiedViews (UV). Použít tento nástroj bylo nutné ze zadání práce. Navíc existuje mnoho výhod, které přináší. Použití UV ušetřilo mnoho hodin práce, navíc lze z jeho diagramů dobře vyčíst, jak převody dat probíhají a zajišťuje možnosti ladění (*angl. debugging)* a plánování běhu (*angl. scheduling).* Později se ovšem ukázaly i některé nevýhody UV, viz níže.

3.1.1 Stručný popis ETL nástroje UnifiedViews

Nástroj označený jako ETL (*angl. extract, transform, load),* nabízí možnosti stahování a transformace dat. ETL nástroj také zajišťuje nahrávání dat na server do databáze nebo do souboru a běžně podporuje další přidružené funkce jako logování, zálohování, ladění transformací, plánování běhu atd. UnifiedViews je ETL nástroj, který běží na dedikovaném serveru a přistupuje se k němu přes webové rozhraní. UV může používat více uživatelů najednou a jeden uživatel může mít najednou spuštěných více různých transformací.

Pro nás byla dostupná studentská instance UV na KSI MFF UK. UV umí extrahovat data z nějakého formátu (např. z HTML, JSON, XML, atd.) do RDF. Dále umí na datech v RDF dělat různé transformace (podporuje většinu SPARQL dotazů včetně *CONSTRUCT*, *UPDATE* a *SELECT*) a posléze je také nahrát na server. S UV se pracuje tak, že se vytvoří tzv. datovod (*angl. pipeline)* a ten se spustí. Datovod je orientovaný graf příkazů, které se mají provést a tyto příkazy mají formu DPU - jednotek zpracování dat (*angl. data processing unit).* DPU je malý Java program a UV jich má mnoho předem připravených. Jedno DPU umí z HTML stránky pomocí CSS selektorů vytáhnout data a převést je do formátu RDF. Jiné DPU umí uložit text do souboru, jiné načítá RDF data ze souboru, další vykonává SPARQL dotaz nad RDF daty, atp. Programátorovi stačí správně pospojovat a nakonfigurovat jednotlivá DPU. DPU se většinou konfigurují tak, že se jim zadá SPARQL dotaz, který mají vykonávat. Hodně nízkoúrovňové práce, jako je navazování spojení na server, stahování souborů pomocí HTTP GET nebo POST včetně ukládání do mezipaměti (*angl. cache),* čtení souborů z disku atd., se tím velmi urychlí, jelikož stačí použít správné DPU. V plánovači UV lze zadat automatické periodické spouštění datovodů. Další výhodou je možnost sledovat historii běhů a zpětně prohlížet ladící data.

Většina DPU na stahování dat používají mezipaměť. Při ladění je velmi vhodné číst data z mezipaměti. Naopak při opravdovém stahování je potřeba mezi-

paměť zakázat, aby se nepracovalo se starými daty. Tok dat datovodem funguje následně. Jedním z DPU se stáhnou data, naparsují se do formátu RDF a pak tečou dál. Záměrně říkáme že tečou, protože pokud jedno DPU dostane za úkol stáhnout např. sto souborů, další DPU je dostane jeden po druhém. Po převodu do RDF pak z každého souboru vznikne jeden RDF graf a ty také tečou dál samostatně. Zde přichází do hry možnost *per-graph execution*, kterou má prakticky každé DPU. Chce-li DPU na každý graf nahlížet zvlášť, použije se *per-graph execution*. Pokud se *per-graph execution* zakáže, všechny grafy na vstupu daného DPU se sjednotí do jednoho. Zde přichází ještě jedna malá komplikace. Pokud je totiž grafů moc, sjednocení skončí chybou. Aby se předešlo této chybě, je potřeba použít speciální DPU zvané *t-graphMerger*. Co se týče více vstupů jednoho DPU, není možné, aby po dvou vstupech zároveň vtékaly izolované grafy. Normálně musí být na všech vstupech pouze sjednocené grafy, tedy na každém vstupu pouze jeden graf (pokud jsou vstupy alespoň dva). Jedinou výjimkou je použití DPU *T-sparqlLinker*, které umožňuje na jednom vstupu tok izolovaných grafů a ostatní musí být zase sjednocené.

3.2 Volba způsobu implementace webové aplikace a volba databáze

Pro tvorbu webové aplikace jsme zvolili klasické webové technologie. Frontend je psaný v HTML5, CSS3 a Javascriptu s použitím knihoven jQuery, Bootstrap a cookie. Backend je psaný v čistém PHP (verze 5.4). Volba webových technologií nebyla těžká, všechno jsou to dobře zavedené, zdokumentované a otestované technologie, které mají velkou komunitu vývojářů. Proto se s nimi dobře programuje a dobře se řeší problémy.

Už méně obvyklé je, že jako databázi, která by poskytovala zdroj dat pro webovou aplikaci jsme zvolili Virtuoso. Důvody pro tuto volbu byly zmíněné v předchozí kapitole 2.1.3.

3.3 Implementace vytěžování a převodů dat

Vytěžování a převody dat byly implementované jako datovody v UV. Jednotlivé datovody sdílí mnoho společných prvků. Většina datovodů obsahuje jedno nebo více testovacích DPU, která ověřují, že se stáhlo všechno a že jsou splněné všechny předpoklady pro zpracování. Taková DPU mají ve svém názvu *TEST*. Více o testovacích DPU a obecně o testování později, ve vlastní kapitole. DPU, která mají v názvu *DEBUG*, slouží k ladění. Tato DPU jsou většinou určená k výběru malého vzorku dat, který pak lze nechat téct datovodem a pozorovat jej.

3.3.1 Získávání vstupu a tvorba výstupu v datovodech

V naší aplikaci bylo nutné umět stáhnout soubory uložené ve formátech HTML a JSON a konvertovat je do RDF. To bylo nutné pro vytěžení dat SVS. Také bylo potřeba umět číst soubory exportovaných výstupů jiných datovodů a umět se přímo dotazovat serveru Virtuoso pomocí dotazovacího jazyka SPARQL. První

bylo nutné pro tvorbu integrované databáze. Druhé bylo vhodné spíše pro ladění datovodů a kontrolu dat. Vše zajišťují malé skupiny správných DPU (červená pro stahování a modrá pro konverze do RDF). Jedná se o *uk-e-HttpDownload-cached*, *UK-T-HtmlCss*; *e-HttpPost*, *t-JsonToJsonLd*, *T-FilesRenamer*, *T-FilesToRdf*; *e-sparqlEndpoint* a *e-filesFromScp*, *T-FilesToRdf*.

Původní myšlenka byla nahrávat výstupy datovodů na Virtuoso a pak se přímo dotazovat pomocí SPARQL, transformovat data a znovu je nahrát na Virtuoso (třeba do jiného grafu). Při analýze Virtuosa jsme ovšem zjistili, že při práci s velkým množstvím dat (deset milionů RDF trojic) není z technických důvodů možné použít přímé dotazování. Virtuoso (i open-source verze 07.20.3217) trpí nedostatkem, že není schopné vrátit příliš velkou odpověď pro dotaz *CONSTRUCT* (viz Williams a Klímek, 2014). Na první pohled je zvláštní, že při dotazu *SELECT* tento problém nenastává. Jedná se údajně o nějaký problém s bufferem. Dotaz *SELECT* ovšem nevrací data ve formátu RDF, proto nešel použít pro získání dat pro tvorbu integrované databáze. Jako důsledek bylo potřeba při tvorbě integrované databáze pracovat s exporty výstupů datovodů a při stahování dat ČOI s exporty datových sad, které jsou na doméně Opendata.cz dostupné.

Co se týče výstupů datovodů, bylo potřeba umět generovat zmíněné exporty a také umět data nahrát na Virtuoso. To druhé bylo potřeba, aby pak Virtuoso mohlo sloužit webové aplikaci jako zdroj dat a také pro ladění datovodů a kontrolu dat. Výstup zajišťovala zase jiná DPU (zelená). Jednalo se o *uv-t-rdfToFiles*, *L-FilesToScp* a *uv-l-filesToVirtuoso Student*.

3.3.2 Přesuny dat při vytěžování a stahování

Při vytěžování, stahování a integraci se data přesouvají mezi různými servery a uvnitř Virtuosa také mezi různými grafy 3.1. Jako poslední dva kroky snad všech datovodů je nahrání exportu jeho výstupu na server pomocí protokolu SCP (*angl. Secure copy)* a nahrání výstupu v RDF na Virtuoso. Naopak jako první krok se často exportovaná data výstupů jiných datovodů čtou a pak se s nimi pracuje. Exporty proto musejí mít unikátní jména, jinak by se přepsaly a ztracená data už by nešla načíst dalším datovodem. Níže je seznam jmen použitých pro exporty.

SVS mapa (webova aplikace)
/data/student−dumps/vypod_svs_mapa_[kod mapy]−[datum stazeni].ttl
(napr. /data/student−dumps/vypod_svs_mapa_E01−2016−06−25.ttl)

SVS registracni tabulky
/data/student−dumps/vypod_svs_reg_[zkrakta tabulky].ttl
(napr. /data/student−dumps/vypod_svs_reg_EU1.ttl)

COI
/data/student−dumps/vypod_coi.ttl

APP nespojene
/data/student−dumps/vypod_appnes_coi.ttl
/data/student−dumps/vypod_appnes_svs.ttl
APP
/data/student−dumps/vypod_app.ttl

TEST (tyto nejsou dulezite)
/data/student−dumps/vypod_test_coi.ttl

/data/student−dumps/vypod_test_svs.ttl
/data/student−dumps/vypod_test_app.ttl

Při nahrávání dat na Virtuoso se data nahrávají do konkrétních grafů. Níže je seznam grafů, které používáme pro skladování dat a také pro integrovanou databázi. První tři obsahují stažená data SVS a ČOI, přičemž jeden slouží pouze pro ladění datovodů při vytěžování dat SVS. Předposlední graf slouží pro uchovávání dat integrované databáze a jako zdroj dat webové aplikace. Poslední slouží pro testování integrace dat a funkcionality webové aplikace.

<http://vypod.cz/svs/debug>
<http://vypod.cz/svs>
<http://vypod.cz/coi>
<http://vypod.cz/app/nespojene>
<http://vypod.cz/app>
<http://vypod.cz/test/app>

Každý, ze zmíněných grafů na Virtuosu svým obsahem odpovídá nějaké množině exportů výstupů datovodů 3.1.

3.3.3 Stahování dat ČOI

Data ČOI, obohacující data ARES a souřadnice stačilo stáhnout. Byl ovšem problém v tom, že dat bylo moc. Proto se data musela stáhnout z exportů datových sad, místo přímého dotazování na Virtuoso, přes které jsou RDF data na Opendata.cz dostupná. Exporty jsou na Opendata.cz také veřejně dostupné (tedy až na export souřadnic kontrol ČOI). Pro tuto výjimku jsme museli data stáhnout do podoby CSV pomocí SPARQL *SELECT* dotazu, dočasně je dát hostovat na HTTP server, odtud je stáhnout, naparsovat a převést znovu do podoby RDF ve správné ontologii. Na velikost výsledku dotazu *SELECT* totiž Virtuoso neklade žádné omezení, na rozdíl od dotazu *CONSTRUCT*.

3.3.4 Vytěžování dat SVS

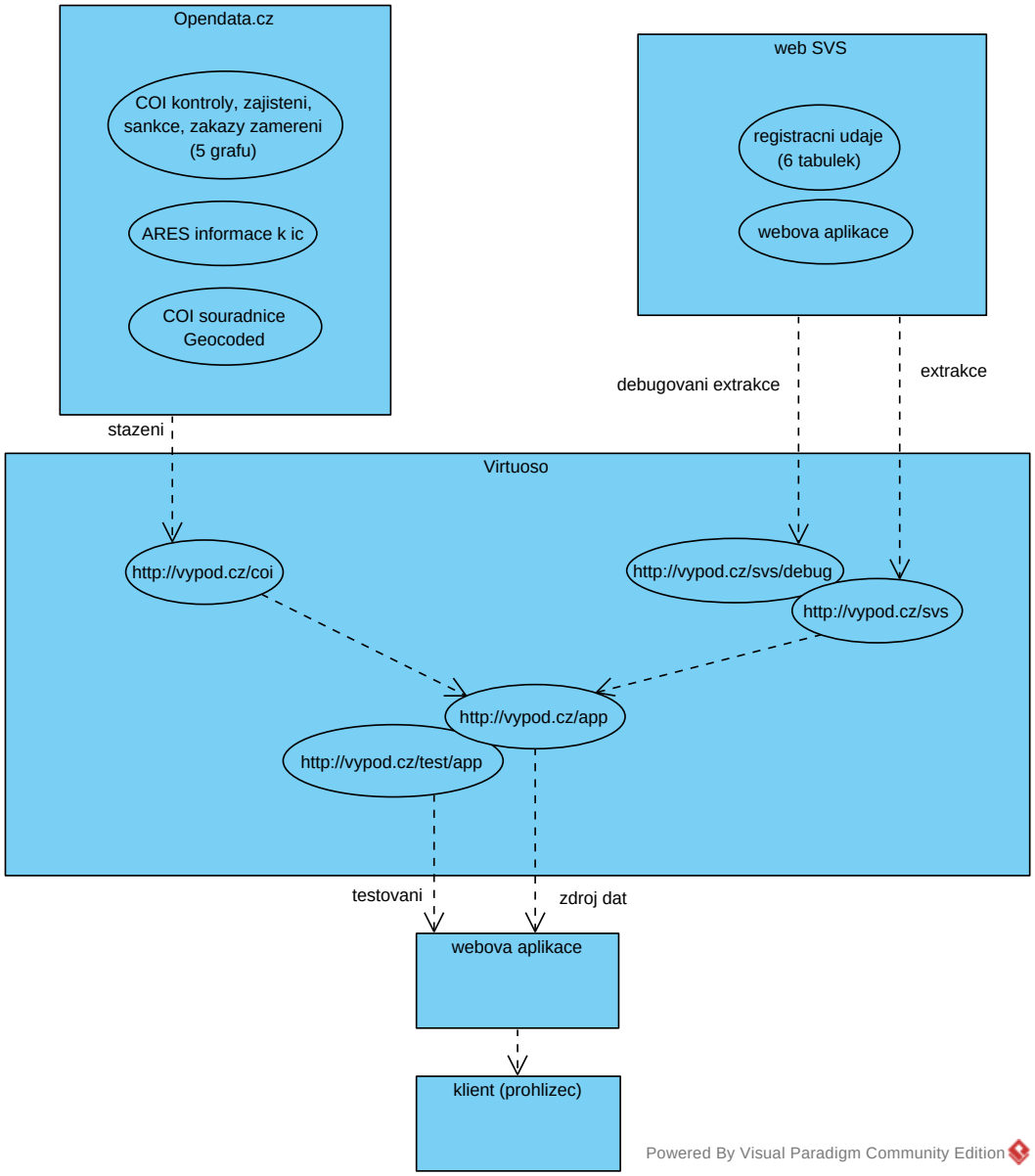
Obecně při stahování registračních tabulek SVS může jakýkoliv atribut chybět, i když se to na první pohled nemusí zdát. Jedinou výjimkou jsou registrační číslo a odkaz na detail.

Všechny datovody pro stahování registračních tabulek 3.2 SVS vykonávají přesně to, co bylo popsáno v návrhu a nakonec nahrají data na Virtuoso do

zkratka	**prefix**	**graf**
SVS	vypod_svs	http://vypod.cz/svs
COI	vypod_coi	http://vypod.cz/coi
APP	vypod_app	http://vypod.cz/app
APP nespojene	vypod_appnes	http://vypod.cz/app/nespojene
TEST	vypod_test	http://vypod.cz/test/app

Tabulka 3.1: Prefixy souborů s exportovanými výsledky z datovodů a jim odpovídající grafy

Každý uvedený graf na Virtuosu odpovídá svým obsahem přesně všem exportům s daným prefixem.



Obrázek 3.1: Přesuny dat při stahování a integraci

zmíněného grafu http://vypod.cz/svs a exportují výstupní data do odpovídajících souborů. Navíc v nich probíhají nějaké další akce jako parsování seznamu druhů chované zvěře, formátování data atp. Jakmile jsou všechny kroky extrakce dat hotové, přichází DPU zvané *aplikuj novou ontologii*, které správně přejmenuje RDF vlastnosti a typy tak, aby výsledek odpovídal navrhované cílové podobě dat SVS. Toto DPU také zajistí spojení všech stejných provozoven, podnikatelů, adresa, atd. z různých registračních tabulek použitím jednotných URI. Navíc toto DPU vytvoří pár nových uzlů, které cílová podoba dat SVS vyžaduje. Výstup tohoto DPU už je finální, data se pak nahrávají na Virtuoso a vytváří se jejich export.

Datovod pro stahování dat z webové aplikace SVS 3.3 dělá to, co bylo popsáno v návrhu. Úseky se stahují po jednom a proto je potřeba před spuštěním ručně nastavit několik parametrů v různých DPU 3.4. Jinak jeho způsob fungování je stejný jako u datovodů pro registrační tabulky. Nejdříve se všechno stáhne tak jak bylo navrženo a pak se data pomocí DPU zvaného *aplikuj novou ontologii* převedou do požadované cílové podoby. Výsledná data se exportují do souboru s unikátním jménem a pak se nahrají na Virtuoso do grafu http://vypod.cz/svs.

Podmínka pro stahování dat z webové aplikace

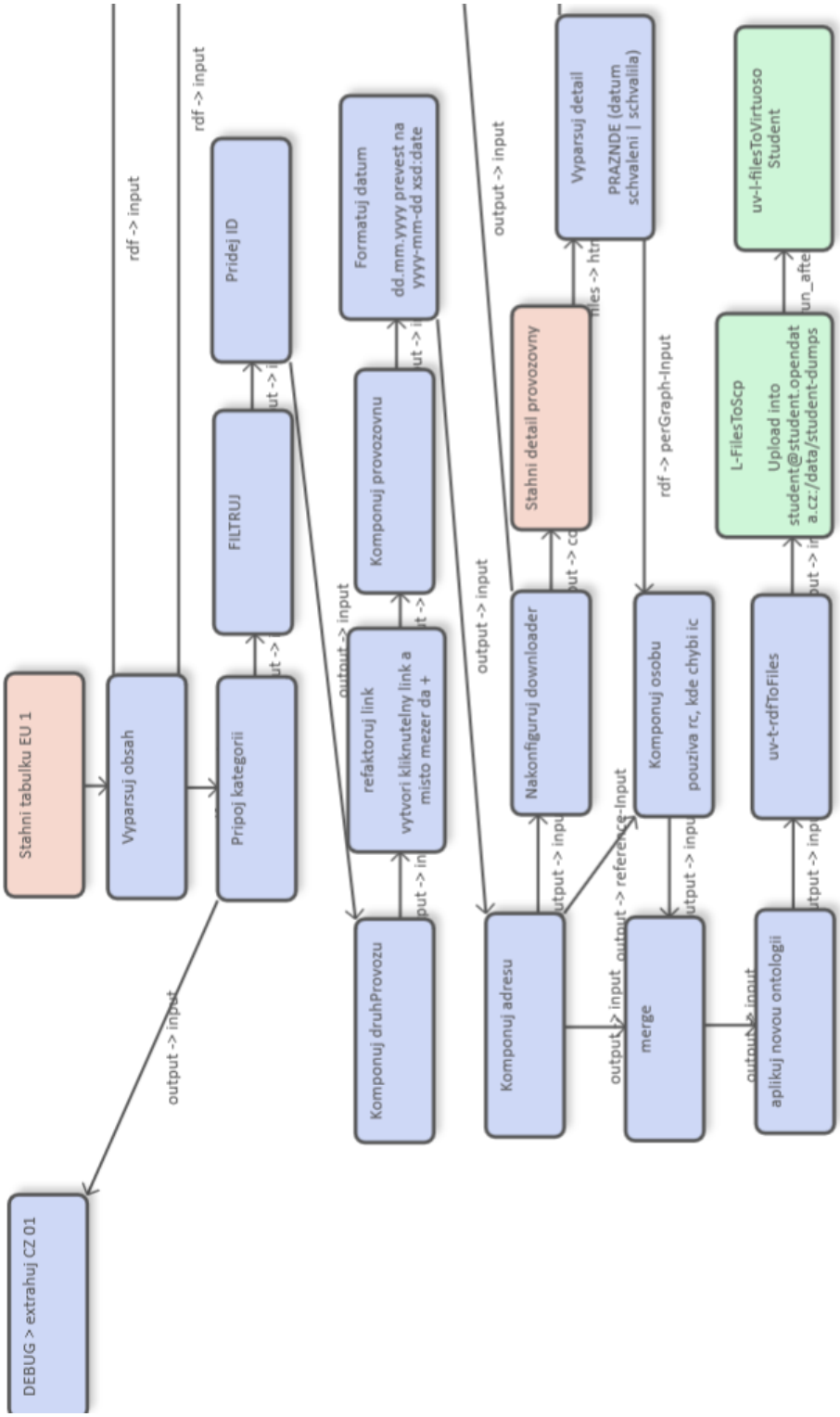
Protože datovod generuje nezávadné kontroly na základě nepřesného data, je potřeba jej správně spouštět. Nesmí se stát, že se stejná data nechají stáhnout dvakrát s různým zadaným datem. Vznikla by duplikace všech nezávadných kontrol v daném úseku. Tato situace by například mohla nastat, kdyby se nejdříve nechaly stáhnout všechny kontroly v tržní síti za březen roku 2016 se zadaným nepřesným datem 2016-03-01 a později by se stáhly všechny kontroly v tržní síti za celý rok 2016 se zadaným datem 2016-01-01.

3.3.5 Časy běhu

Při používání UV i obecně při stahování dat po síti nelze přesně měřit dobu stahování a spoléhat, že příště stahování potrvá stejně dlouho. Níže je uvedeno, jak dlouho trvaly poslední běhy všech datovodů 3.2 (pro přibližnou orientaci).

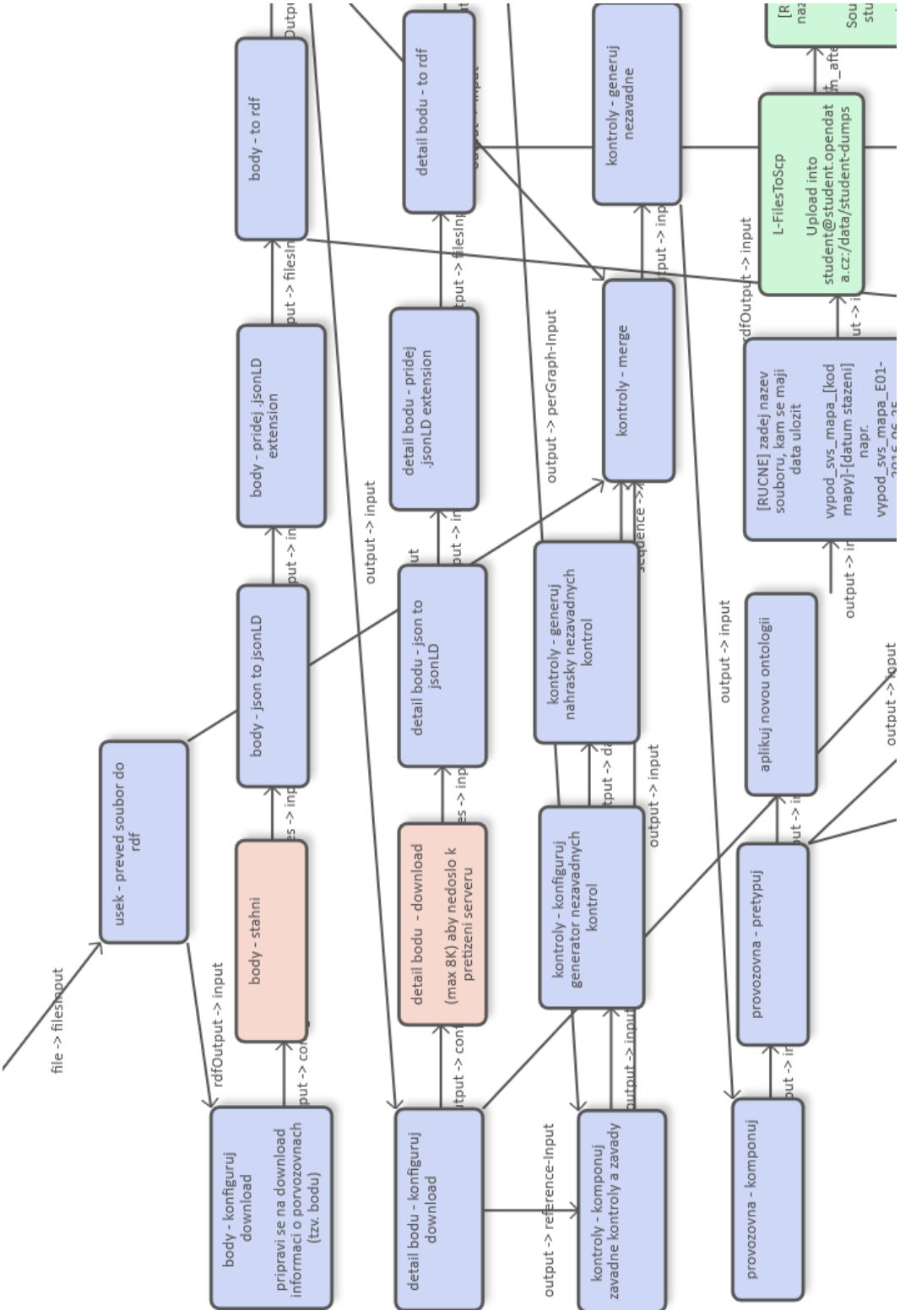
3.3.6 Obecné problémy při vytěžování dat z webu

Nápad extrahovat data z HTML stránky má na první pohled hned několik důvodů se nepovést. Za prvé se musí počítat se změnou HTML/PHP kódu stránky a také jejího obsahu. To se nám stalo v půlce práce, kdy přišla změna struktury všech registračních tabulek a bylo potřeba předělat všech šest datovodů. Za druhé se musí počítat s tím, že kód stránky obsahuje chyby a tudíž nemusí být možné vytěžit všechna data z jednoduchého důvodu, že na stránce ani nejsou zveřejněná, i když by měla být. Stránky SVS obsahovaly několik chyb - od invalidních JSONů až po špatné PHP, které bránilo stahování detailů o podnikatelích (hlavně IČ). Hlavní chyby byly naštěstí postupně odstraněny po kontaktování technické podpory. Za třetí je potřeba počítat s tím, že není žádná garance. Nám se stalo, že v půlce práce SVS smazalo všechna data o kontrolách z roků 2013 až 2015 a začalo publikovat pouze data od roku 2016. To byl velmi nečekaný a nepochopitelný tah. Za čtvrté je potřeba počítat s tím, že stránka bude nečekaně



Obrázek 3.2: Vytěžování dat SVS z registrační tabulky EU 1

Datovody pro vytěžování ostatních registračních tabulek vypadají podobně, ovšem ne úplně stejně.



Obrázek 3.3: Vytěžování dat SVS z jejich webové aplikace

Obrázek 3.4: DPU, která je potřeba (jednoduše) ručně konfigurovat před spuštěním vytěžování dat z webové aplikace SVS

na několik dní mimo provoz, ať už kvůli údržbě databáze nebo něčemu jinému. Za páté je naopak potřeba očekávat, že chyby v datech a ve webových stránkách budou postupně opravovány, tudíž jakákoliv opatření na obejití specifických chyb při stahování budou jedno po druhém přestávat být potřebná a v horším případě začnou být na škodu.

3.4 Integrace dat ČOI a SVS

Integraci dat ČOI a SVS zajišťují tři myšlenkově jednoduché datovody - *IN COI to APP*, *IN SVS to APP* a *IN APP to APP*. První stáhne export dat ČOI odpovídající grafu http://vypod.cz/coi ve Virtuosu, převede data do modelu APP dle návrhu, výsledek datovodu exportuje do souboru *vypod_appnes_coi.ttl* a také ho nahraje na Virtuoso do grafu http://vypod.cz/app/nespojene.

Druhý dělá to samé s daty SVS, akorát vstupní data nejsou v jednom exportu, ale v několika a jeho výsledek se uloží jako export *vypod_appnes_svs.ttl*. Stahuje tedy všechny exporty, které dohromady svým obsahem odpovídají obsahu grafu http://vypod.cz/svs ve Virtuosu. To jsou všechny s předponou *vypod_svs*. V datovodu *IN SVS to APP* je velmi důležité nezapomenout sjednotit data z jednotlivých exportů do jednoho velkého grafu (což datovod dělá). Kdyby to ovšem

datovod	**čas**	**počet trojic**
EU 1	1 h 50 min	42 000
EU 2		1 000
EU 10	2 h 50 min	70 000
OO 5	5 min	3 000
PP 1	3h 40min	90 000
EU 4	1h*	10 000*
MAPA vejce		500
MAPA vejce, balení		1 000
MAPA ryby		1 500
MAPA králíci		500
MAPA mlékarny	5 min	6 000
MAPA med		1 000
MAPA dovoz	15 min	20 000
MAPA masné výrobny	10 min	10 000
MAPA drůbeží porcovny		2 000
MAPA drůbeží		500
MAPA jatka	10 min*	10 000
MAPA tržní síť	1 h 5 min	32 000
MAPA zvěřina		800
MAPA bourárny	15 min	8 000

Tabulka 3.2: Časy běhů datovodů a počty stažených RDF trojic při extrakci dat SVS

MAPA je jen jiný název pro webovou aplikaci SVS. Zde jsou záznamy ze dne 26.6.2016 a stahují se všechny dosavadní kontroly za rok 2016. Kde nejsou uvedené časy, znamená to, že doba běhu datovodu je kratší než pět minut. (* - jedná se o nepřesné odhady, kdy se čas nebo počet hran nepodařilo dobře změřit)

nedělal, běžel by pár hodin a pak by skončil výjimkou *OutOfMemoryError: Java heap space*. Jedná se nejspíš o chybu v UV, jelikož není žádný opodstatněný důvod pro takové chování.

Poslední datovod zajišťuje spojení výsledků předchozích dvou. Stáhne exporty jejich výsledků a data spojí dle návrhu. Aby správně proběhlo odstraňování duplikací, stáhne také data z aktuální integrované databáze. Tato data se nacházejí v exportu *vypod_app.ttl*. Výsledek uloží do toho samého exportu a také data nahraje na Virtuoso do grafu http://vypod.cz/app. Obsah tohoto grafu již představuje integrovanou databázi připravenou k použití.

3.4.1 Ladění integrace a testování

Pro testování integrace je na Virtuosu vyčleněný speciální graf pojmenovaný http://vypod.cz/test/app. Od každého datovodu pro integraci existuje jeho testovací verze (její název místo *IN* začíná *TIN*), jejíž jádro je stejné, ale zbytek je jednodušší. Nepotřebuje totiž být stavěná na velké množství dat jako originály. K testování se nepoužívají soubory s exporty, ale jednoduše datovody *TIN COI to APP* a *TIN SVS to APP* nahrají data do grafu http://vypod.cz/test/app a pak se použije *TIN APP to APP* na spojení. Zde nepoužíváme dva grafy - jeden pro nespojená data a druhý pro spojená - protože to není potřeba. Zatímco v netestovém grafu http://vypod.cz/app vždy máme jen správně spojená data, proto se navíc používá také graf http://vypod.cz/app/nespojene jako mezikrok.

3.5 Jak integrovat data z nového zdroje

Pro přidání dat z nového zdroje je potřeba vyhradit nové jméno pro soubor s exportem. Jméno musí být ve tvaru vypod_appnes_[cokoliv].ttl. Data z nového zdroje je potřeba převést do modelu APP a výsledek převodu exportovat do zmíněného souboru. Případně lze navíc nahrát data na Virtuoso do grafu http://vypod.cz/app/nespojene pro ladění (tak, jak to dělají naše datovody). Pak už stačí jen spustit datovod *IN APP to APP*, který jako vstup používá všechny soubory ve zmíněném tvaru.

Pro ladění je lepší si nejdříve vše vyzkoušet na testovacím grafu, stejným způsobem jako jsme my ladili integraci dat ČOI a SVS.

Přesné technické detaily pro vytváření exportu na správném serveru a nahrávání dat do Virtuosa zde z bezpečnostních důvodů neuvádíme. Postup je potřeba nastudovat z nějakého z našich datovodů.

datovod	**pracovní název**	**čas**	**počet trojic**
stazeni COI	EC	2 h 50 min	28 700 000
COI do APP	COI to APP	1 h 15 min	1 021 369
SVS do APP	SVS to APP	1 min	200 000
APP do APP	APP to APP	6h 45 min	915 282

Tabulka 3.3: Časy běhů datovodů a počty stažených RDF trojic při integraci a stahování dat ČOI

3.6 Implementace webové aplikace nad integrovanými daty

Pomocí zvolených technologií jsme podle návrhu implementovali webovou aplikaci, která jako zdroj dat používala integrovanou databázi dostupnou na SPARQL serveru Virtuoso. Bylo potřeba zvolit způsob komunikace s databází a implementovat backend včetně práce se souřadnicemi (za použití funkcí Virtuosa). Také bylo potřeba implementovat responzivní frontend s použitím Google Maps API a knihoven Bootstrap a jQuery.

Pro běh aplikace je potřeba zajistit HTTP server s verzí PHP alespoň 5.6. Dále je nutné, aby databázový server KSI MFF UK (Virtuoso) na adrese `http://student.opendata.cz/sparql` byl online a nikdo z administrátorů z něj nesmazal data. Nakonec nesmíme zapomenout na to, že kód HTML používá síť CDN (*angl. Content Delivery Network)* pro stažení knihoven Bootstrap a jQuery. Servery, které CDN zajišťují musí být dostupné a musí stále hostovat správnou verzi této knihovny. Navíc i služba Google Maps API musí být dostupná. Je mizivá šance, že by zde nastal problém, ale i přesto je dobré o tom vědět.

3.6.1 Práce s databází

Pro práci s databází jsme zvolili PHP knihovnu Semsol/Arc2, ale posléze jsme od ní museli upustit. Pracovalo se s ní sice příjemně, ale měla zásadní vadu. Nepodporovala verzi 1.1 jazyka SPARQL (pouze 1.0) a kvůli tomu s ní nešly používat pod-dotazy (*angl. subqueries),* které jsme v naší aplikaci potřebovali pro rozumnou implementaci dotazů a snížení datového provozu po síti. Namísto této knihovny jsme přešli k manuálnímu dotazování. To znamená posílání SPARQL dotazů jako parametry HTTP dotazu *GET* a následné parsování odpovědi ve formátu JSONu.

3.6.2 Převod odpovědí Virtuosa z JSON do PHP

Jako formát odpovědí SPARQL databáze (přístupového bodu) jsme zvolili JSON, jelikož se nejlépe parsuje a zpracovává. Níže je příklad dat vrácených Virtuosem na SPARQL dotaz s požadovaným formátem JSON.

```
{
  "head": { "link": [], "vars": ["s", "p", "o"] },
  "results":
  {
    "distinct": true,
    "ordered": false,
    "bindings":
    [
      {
        "auto": { "type": "uri", "value": "http://example.cz/uzel/auto/1" },
        "znacka": { "type": "uri", "value": "http://example.cz/Mercedes" }
      },
      {
        "auto": { "type": "uri", "value": "http://example.cz/uzel/auto/2" },
        "znacka": { "type": "uri", "value": "http://example.cz/Citroen" }
      }
    ]
```

```
  }
}
```

Následovně budeme v PHP schopni získat první řádek z výsledku ve formátu JSON:

```
$decoded_json['results']['bindings'][0]['auto']
$decoded_json['results']['bindings'][0]['znacka']
```

3.6.3 Struktura kódu backendu

Adresářová struktura kódu webové aplikace je následující. Úvodní stránka (index.php) je v kořenovém adresáři a každá další stránka má svou vlastní podsložku ve složce *src* (pro přehlednost). Co se týká externích knihoven, některé jsou ve složce *lib* a na ostatní je pouze odkaz někde v kódu, protože jsou poskytované sítí CDN.

Všechny SPARQL dotazy, kterými aplikace získává data z databáze, jsou ve složce *spql* a jsou parametrizovatelné pomocí maker. Třída *Sparql.php* obsahuje všechny funkce pro práci s databází a zpracovávání maker. Níže je příklad makrovatelného dotazu.

```
PREFIX s: <http://schema.org/>
PREFIX adms: <http://www.w3.org/ns/adms#>

SELECT ?ico ?nazev ?adresa ?kraj ?mesto ?ulice ?psc
FROM <GRAF>
WHERE {

        <PROVOZOVNA> a s:LocalBusiness ;
                s:parentOrganization ?osoba .

        ?osoba a s:Organization ;
                adms:identifier ?ico .

        <ICO>
}
```

Za makra je pak možné dosadit libovolný vhodný kus SPARQL kódu. Níže je příklad platného dosazení za makra.

```
<GRAF> := <http://vypod.cz/app>
<PROVOZOVNA> := <http://vypod.cz/node/provozovna/CZ+123423>
<ICO> := FILTER(?ico = '00125432')
```

Složka *test* obsahuje testy PHPunit, které budou detailněji popsané ve vlastní kapitole.

Důležitá je statická třída *GLOB*. Jedná se o jakousi globální třídu v tom smyslu, že ji používá bezmála každý soubor. GLOB obsahuje důležité přepínače a sbírku statických metod vhodných pro použití prakticky kdekoliv. Hlavní přepínače rozhodují, zda je stránka v režimu testování webu, testování databáze nebo v normálním režimu. V prvním případě se využívají vymyšlená testovací data a nepracuje se s databází. V druhém případě se používá testovací databáze naplněná jinými testovacími daty. Přepínač *UDRZUJ_NA_ZIVU* zajišťuje, že interpretace kódu PHP neskončí při první chybě, ale bude pokračovat co nejdéle.

Všechny chyby, které nastaly, se evidují a nakonec se zobrazí v patičce stránky. Hlavní problém, který třída *GLOB* řeší, je zajištění robustního mechanizmus pro komplikovanější inkluze pomocí funkce *GLOB::incl()*. Komplikovanější inkluze s použitím relativních cest totiž v PHP celkem rychle vyústí v chybu typu "soubor neexistuje", i když soubor doopravdy existuje. *GLOB:incl()* umožní inkluzi souborů stejným způsobem jako je běžné v HTML - pomocí absolutní cesty z kořene webu.

U ostatních tříd a funkcí je z kódu čitelné jak fungují a co dělají.

3.6.4 Přesné fungování práce se souřadnicemi ve Virtuosu

Souřadnice GPS jsou body na kouli vyjádřené pomocí úhlů. Naopak souřadnice na mapě jsou body v rovině. Z koule lze souřadnice zobrazit do roviny a naopak. Ovšem různých druhů projekce je mnoho a každý má své výhody a nevýhody, co se týká zkreslení vzdáleností, úhlů atp. Nejpoužívanější je Merkátorova projekce. Její modifikaci (webovou Merkátorovu projekci) používá Google při kreslení mapy na obrazovku.

K určení, o jaký druh souřadnic se jedná, se používá tzv. SRID (viz Tengshe, 2013). SRID pro GPS souřadnice je 4326 podle standardu WGS84. Google jim také říká hodnoty zeměpisné šířky a délky (*angl. latitude and longitude values)* (viz Google, 2016b). SRID pro souřadnice v rovině je většinou 3857. Google jim říká světové souřadnice (*angl. world coordinates).*

Důležité je, že to co zobrazují Google Maps, to s čím pracujeme při používání Google Maps API a to co vrací Google Geocoding API, jsou souřadnice se SRID 4326.

Vzdálenost středů dvou měst na Zemi se počítá prakticky přesně jako délka oblouku mezi jejich GPS souřadnicemi. K tom slouží tzv. Haversinova funkce (*angl. Haversine formula).*

Ve Virtuosu existuje specifická funkce *bif:st_contains(x, y, r)*, která spočítá Haversinovu vzdálenost mezi body x a y a vrátí *true*, pokud jsou k sobě blíže než r kilometrů. Musí ovšem být zajištěno, že SRID bodů je 4326, tedy jedná se o GPS souřadnice. Pokud tomu tak není, počítá se normální vzdálenost v rovině a body se neinterpretují jako souřadnice na kouli, jedná se tedy o něco úplně jiného.

Funkce specifické pro Virtuoso lze použít ve SPARQL dotazu na koncový bod poskytovaný Virtuosem a budou fungovat, i když nejsou součástí standardu SPARQL. Takové funkce začínají prefixem *bif:*. Všechny prefixy se normálně deklarují na začátku SPARQL dotazu, ale u *bif:* je výjimka a není to potřeba. Funkce pro práci s prostorovými daty ve Virtuoso začínají své jméno prefixem *bif:st_*.

Je důležité si uvědomit, že při vytváření bodu ve Virtuosu pomocí *st_point(x, y)* je potřeba uvést jako první argument zeměpisnou délku (*angl. latitude)* a jako druhý zeměpisnou šířku (*angl. longitude)* (viz OpenLink Software, 2016b). Což je přesně naopak, než jak je zvykem souřadnice zapisovat.

Při zadávání souřadnicových dat do Virtuosa, tak aby se na ně Virtuoso dívalo jako na prostorový bod, je nestačí vložit jako text. Je potřeba použít formát WKT a otypovat literál jako *vir:Geometry*. Střed Prahy by tedy odpovídal POINT(14.4380030, 50.074426)^^vir:Geometry. Zde, stejně jako u *bif:st_contains*, se zase uvádí nejdříve zeměpisná délka a posléze zeměpisná šířka. Subjekt k sobě

souřadnicová data (*angl. geo data)* většinou připojuje predikátem *geo:geometry*, ale nemělo by to být povinné. Takto zadaný bod bude ve Virtuosu pochopený jako prostorový bod a budou pro něj fungovat funkce *bif:st_contains(x, y, r)* a další, ale to není vše. Navíc Virtuoso takovýto bod automaticky přidá do prostorového indexu (R-stromu) a dotazy používající např. *bif:st_contains(x, y, r)* jej budou automaticky využívat. Dojde tak k obrovskému zrychlení na velkých datech. Níže jsou plná znění použitých zkratek.

vir: http://www.openlinksw.com/schemas/virtrdf#
geo: http://www.w3.org/2003/01/geo/wgs84_pos#

3.6.5 Principy Bootstrapu

Bootstrap je CSS a JS knihovna poskytující moderní vzhled a responzivitu HTML stránek. JS používá převážně pro zajištění správných designových vlastností, tedy pouze jako pomocníka CSS. Bootstrap definuje mnoho tříd a jejich CSS styly. Používá se přidáním třídy ke správné HTML značce. Aby vše fungovalo správně, musí být třídy přidané ke správným značkám a ve správných kombinací. Mezi důležité třídy patří *.container*, *.row*, *.col-lg-2*, atd.

Responzivitu zajišťuje klasická technika dvanáctimístné mřížky (*angl. grid).* Stránka se rozdělí na řádky (*.row*) a každý řádek se rozdělí na různě široké sloupce, které dohromady zabírají dvanáct míst. Např. *.col-lg-2* je sloupec zabírající dvě místa na velkém displeji. Kdybychom chtěli, aby tento sloupec vypadal jinak na menších displejích, mohli bychom jej rozšířit pomocí *.col-lg-2 .col-md-4*. Na větších displejích by zabíral dvě místa a na středně velkých čtyři. Níže je přehled toho, co je myšleno velkým, středním, malým a mini displejem.

xs − for the smallest screen widths like smartphones < 768 px
sm − for small screen widths like smartphones and tablets >= 768 px
md − for medium screen widths like tablets and laptops >= 992 px
lg − for large screen widths like desktops >= 1200 px

Bootstrap mimo jiné také optimalizuje stránku pro čtečky displejů, pokud je správně použitý.

3.6.6 Principy jQuery

JQuery je JS knihovna, která velmi zpříjemňuje práci s JS. Její hlavní výhoda je robustní hledání HTML značek a také mnoho pohodlných funkcí. Hledání značek probíhá pomocí konstruktu označeného dolarem, který přijímá CSS3 selektory a vrací odpovídající jQuery objekty (např. *$('#tlacitko')* by vrátilo jQuery objekt HTML značky s id *tlacitko*). JQuery objekt odpovídá DOM objektu, navíc na něm lze volat mnoho pohodlných jQuery funkcí.

3.6.7 Struktura kódu frontend

Na stránce *index.php* zajišťuje všechny reakce na pokyny uživatele objekt *Mediator*. *Mediator* v sobě uchovává aktuální kriteria, která uživatel zvolil pro vyhledávání. Obsah *Mediatoru* se po odchodu ze stránky zapisuje do cookies a při načítání stránky se z cookies zpátky získává (pokud možno). Základní přínos je

v situaci, kdy uživatel něco vyhledá, zobrazí se mu výsledky a pak se ve vyhledávači vrátí o krok zpět, aby pozměnil parametry. Díky cookies nebude muset znovu zdlouhavě zadávat parametry hledání. Ostatní funkcionalita frontendu je jednoduše pochopitelná.

3.7 Komentáře

Při práci jsme se setkali s různými technickými potížemi a překvapeními. Níže je seznam věcí, na které je potřeba si dát pozor při práci s jednotlivými technologiemi. Komentovali jsme hlavně nástroj UnifiedViews, specifikaci SPARQL a server Virtuoso.

3.7.1 Komentáře k Virtuosu

Pro Virtuoso se těžko programuje, protože o něm nejsou skoro žádné články ani nemá téměř žádnou aktivní komunitu. Navíc obsah dokumentace Virtuosa je prakticky skrytý před roboty webových vyhledávačů (dotazy přes Google a Yahoo ohledně funkcí Virtuosa nevrací článek z dokumentace, ale přinejlepším nějaké odkazy na různá fóra). Tudíž, pokud chceme něco zjistit, je potřeba předem znát název funkce kterou hledáme a vyhledat ji na webových stránkách Virtuosa pomocí jejich textového vyhledávače. Druhá možnost je číst dokumentaci jako knížku, což je potřeba, nezná-li programátor předem název funkce, kterou potřebuje.

Co se týká fungování SPARQLu, je nečekané, že proměnné svázané pomocí BIND([vyraz] AS ?promenna) ve SPARQL dotazu, musí být deklarované popořadě. Tedy uvedený kód, kde bychom čekali ohodnocení proměnných *?a* i *?b* číselnou hodnotou pět, nebude fungovat. Místo toho vznikne chyba *Variable 'a' is used in the query result set but not assigned*. Možná se jedná o obecnou vlastnost SPARQL, nikoliv jen Virtuosa.

```
select ?a ?b
where {
        ?s ?p ?o
        BIND(?a AS ?b)
        BIND(5 AS ?a)
} LIMIT 1
```

Dále z nějakého důvodu nefunguje porovnávání (kalendářních) dat otypovaných jako *xsd:date* (i když by podle specifikace SPARQL mělo). Řešením je převod na řetězce do tvaru yyyy-mm-dd a porovnávání řetězců.

Nakonec jsme u Virtuosa ještě objevili specifický problém při konkatenaci českých řetězců, detaily viz příloha.

3.7.2 Komentáře k UnifiedViews

Nástroj UnifiedViews byl sice velmi užitečný, ale měl několik závad, které zhoršovaly práci. Nejproblematičtější byly časté výpadky a zasekávání, které musel řešit administrátor nástroje restartem. Bohužel se administrátorům nepodařilo přijít na důvod výpadků, takže probíhaly po celou dobu práce. Níže je seznam problémů, na které jsme narazili při této práci.

- Automatické ukládání datovodů před spuštěním funguje jen v polovině případů. V druhé polovině případů vznikne chyba, která zruší celé sezení a je potřeba se znovu přihlásit do UV.
- GUI je dost pomalé - hlavně pro větší datovody (více než 20 DPU). Reakce na každé kliknutí pak trvá průměrně jednu vteřinu.
- UV nepodporuje validaci SPARQLu při zadávání SPARQL příkazů. O chybné syntaxi se dozvíme až po spuštění.
- GUI UV nepodporuje klávesové zkratky (ani Enter, pro uložení konfigurace DPU).
- Dokumentace k DPU (popis toho co dělají a jak se používají) většinou zcela schází. Někdy je také chybná. Na fungování mnoha DPU bylo potřeba přijít odhadem a metodou pokusu a omylu.
- Při parsování více souborů v příslušném DPU nelze dohledat zdroj výjimky způsobené chybou formátu.
- V ladicím režimu při nahlížení dat pomocí dotazu SPARQL *SELECT* při syntaktické chybě v dotazu vznikne chyba UV a zruší se celé sezení. Dokonce se zdá, že to nastane i pro složitější validní dotazy.
- U velkých datovodů (více než 30 DPU) docházelo k fenoménu jakési mrtvé zóny GUI. Pokud se nějaká DPU zatáhla až moc daleko od levého horního rohu obrazovky, ztmavl jejich stín a přestaly být funkční. Nešlo je editovat, mazat, ani přemísťovat. Tomuto problému přidal na závažnosti fakt, že u větších datovodů se při kopírování nové DPU objeví úplně jinde než původní. Kopie DPU se tvořila cca dvakrát níže od vrchu obrazovky něž originál, čímž se u velkých datovodů jednoduše dostala do mrtvé zóny. V jednom momentu jsme málem přišli o několik desítek hodin práce kvůli jednomu DPU v mrtvé zóně, které nešlo smazat a blokovalo spuštění datovodu, protože nemělo přidělený žádný vstup (většina DPU požaduje vstup). Problém se nakonec vyřešil díky tomu, že mrtvá zóna není vždy stejná a po několika desítkách otevření a zavření prohlížeče a provádění různých změn se ji naštěstí podařilo zvětšit dost na to, aby DPU z problémových zón šla smazat.
- V UV chybí dekompozice - nelze vytvořit datovod a spouštět jej jako podprogram v jiných datovodech. Přidání dekompozice by ulehčilo práci a redukovalo chybovost v uživatelských datovodech.
- UV nepodporuje tabulátor při zadávání SPARQL dotazů do určených DPU (stejně jako jej nepodporují textová pole v HTML). Zde ovšem důsledek je takový, že psaní všech SPARQL dotazů musí probíhat v nějakém editoru mimo UV.

Běhové problémy UV:

- Několikrát se UV stalo nepřístupným s HTTP chybou 502. Vyřešil to až restart administrátorem.
- Několikrát bylo UV sice přístupné, ale při spouštění datovodů vracelo hlášku "Backend offline"a nespustilo je. Vyřešil to až restart administrátorem.
- Několikrát se datovod zasekl - byl označený jako běžící, ale celé hodiny nevypsal ani řádku do logu a nešel zastavit. Vyřešil to až restart administrátorem.

- Několikrát se zdálo, že se datovod zasekl, ale doopravdy už byl jeho běh dávno hotový. Vyřešilo to spuštění toho samého datovodu znovu. Spuštění jiného datovodu nemělo žádný efekt.
- V neposlední řadě jednou administrátor bez varování smazal všechny datovody.

3.7.3 Komentáře ke Sparql

Na psaní Sparql dotazů by se hodilo nějaké SPARQL IDE s analýzou syntaxe, podbarvením, nějakou statickou analýzou (sledováním chyb v názvech proměnných) a zkracováním URI podle seznamu prefixů. Takové IDE zatím neexistuje, protože SPARQL a RDF jsou velmi málo rozšířené. My jsme při vývoji museli pracovat s obecným textovým editorem (Notepad++) a online validátorem SPARQLu (viz www.sparql.org, 2016). Tento validátor byl velmi užitečný, ale je potřeba vědět o tom, že i on má své chyby. Například nerozumí dotazům *UPDATE*, takže je považuje za chybu syntaxe. Dále někdy špatně kontroluje použití středníku a tečky, které mají ve SPARQL speciální pravidla. Je tedy možné, že i dotaz označený validátorem za validní nebude fungovat.

4. Testování

Současně s implementací extrakce a integrace dat SVS a ČOI a implementací webové aplikace bylo potřeba nasadit vhodné testy a zařídit, aby prošly. Testovali jsme správnou extrakci všech dat SVS z jejich webových stránek. Také jsme testovali, že integrace dat proběhla správně a v neposlední řadě jsme testovali funkcionalitu naší webové aplikace.

4.1 Při vytěžování dat SVS

Přímo v datovodech pro vytěžování dat SVS se testuje, zda se stáhla všechna chtěná data. Dopředu se počítalo se s nějakou malou neúplností způsobenou chybami v datech nebo v kódu SVS. Pro testování korektnosti existuje vlastní datovod.

4.1.1 Testování úplnosti při vytěžování registračních tabulek SVS

Každý datovod, který vytěžuje registrační tabulky SVS obsahuje dvě DPU, která testují úplnost stažených dat 4.1. Bylo třeba si uvědomit, že prakticky libovolný atribut mohl chybět a podle toho zavést rozumné testy.

DPU *TEST 1* testuje zda se stáhly všechny řádky a zda měly uvedené registrační číslo a odkaz na detail. Pokud tento test neprojde, znamená to jednu ze dvou věcí. První možnost je, že jde o chybu v datech SVS (chybí registrační číslo), tehdy nemá smysl ostatní data stahovat. Registrační číslo je nejdůležitější atribut v datech SVS, podle něj se jednoznačně identifikuje provozovna. Pokud by chybělo, dá se s jistotou usuzovat, že ostatní data o dané provozovně jsou také chybná. Druhá možnost je, že chybí odkaz na detail, tehdy nevadí, že se data nestáhnou, protože se stejně nelze dozvědět, kdo je vlastníkem dané provozovny. Data tedy přestávají být zajímavá.

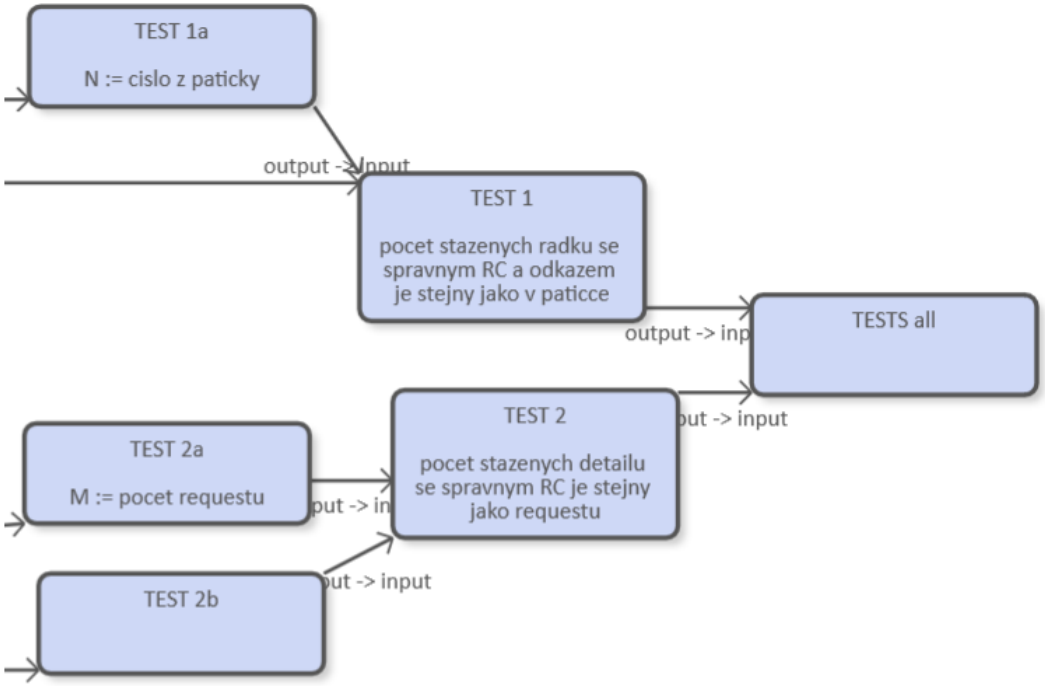
DPU *TEST 2* testuje zda se stáhly všechny detaily provozoven a zda mají uvedené registrační číslo. Pokud tento test neprojde, jde o stejnou situaci jako u předchozího testu.

Pokud některé testy selžou a zjistí se například jeden špatný řádek z tisíce, není potřeba se tím zabývat. Velmi pravděpodobně šlo o chybně zadaná data nebo špatně smazaný záznam v databázi SVS. Kdyby ovšem bylo velké procento špatných řádků, bylo by potřeba zjistit příčinu a nějak se s tím vypořádat.

Jelikož tabulka *EU 4* se stahuje trochu odlišným způsobem než ostatní, i její testy mají jiný význam. První test sleduje počet správně stažených produkčních podniků a druhý potom testuje počet správně stažených rybníků, které daný produkční podnik zastřešuje.

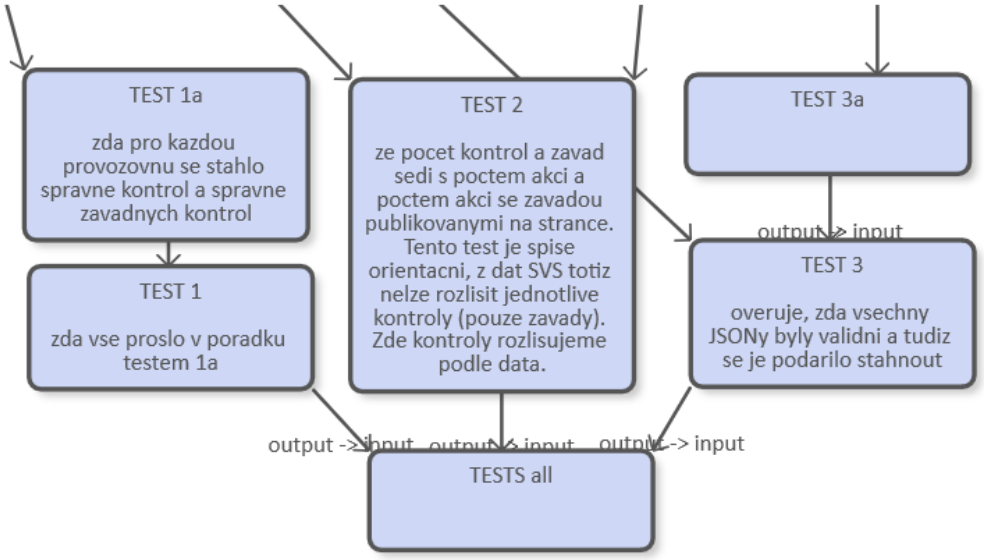
4.1.2 Testování úplnosti při vytěžování dat z webové aplikace SVS

Datovod pro vytěžování dat z webové aplikace SVS obsahuje tři DPU, která testují úplnost stažených dat 4.2.



Obrázek 4.1: Část datovodu, která testuje úplnost vytěžování dat SVS z registrační tabulky EU 1

U ostatních tabulek je tato část stejná, s výjimkou EU 4, kde je podobná.



Obrázek 4.2: Část datovodu, která testuje úplnost vytěžování dat z webové aplikace SVS

DPU *TEST 1* na základě metadat ohledně detailů bodů testuje, zda se pro každý bod (provozovnu) stáhly všechny závadné i nezávadné kontroly. Malé selhání tohoto testu (1 kontrola z 10 se nestáhla) by pravděpodobně znamenalo nepřesnost v datech SVS nebo chybějící záznam o závadné kontrole. Ten by mohl vzniknout chybným zadáním dat. Větší nepřesnost by mohla znamenat chybu v mechanizmu stahování a potřebovala by vyšetřit.

DPU *TEST 2* na základě metadat ohledně seznamu všech bodů v daném úseku testuje, zda se stáhly všechny závadné i nezávadné kontroly v daném úseku. Selhání tohoto testu způsobené selháním prvního testu by bylo v pořádku. V opačném případě by to znamenalo, že se nestáhly všechny body (data všech provozoven v daném úseku). To by byl problém, a proto nastupuje *TEST 3*.

DPU *TEST 3* je pro případ, že selže *TEST 2* a ověřuje, zda se stáhly všechny detaily bodů (JSON soubory). Pokud totiž soubor ve formátu JSON není validní, přeskočí se a datovod běží dál. Pokud *TEST 3* selže, je jasné, že *TEST 2* selhal kvůli tomu, že některé JSONy nebyly validní. S tím se nedá nic dělat a pokud je jich relativně málo, lze to přijmout.

Pokud by ovšem selhal *TEST 2*, ale přitom *TEST 1* a *TEST 3* prošly, znamenalo by to nějakou nečekanou událost a bylo by potřeba ji prošetřit.

4.1.3 Testování korektnosti při vytěžování dat SVS

Korektnost stahování dat SVS testujeme na principu vstupu a očekávaného výstupu. Jelikož by ovšem bylo neúměrně složité tvořit umělé vstupy (bylo by potřeba navíc zařídit vlastní server a imitaci webové aplikace SVS), bereme jako vstup nějaký vzorek dat SVS. Z každé registrační tabulky je vybraná jedna provozovna (a její majitel) a z webové aplikace je vybráno pár bodů (a všechny jejich kontroly a závady). Datovod *ES Testovani korektnosti* pak testuje, zda vytěžená data SVS obsahují dané vzorky a všechny atributy, které se měly vytěžit. Přičemž vzorky byly ručně převedené do požadované cílové podoby dat SVS.

4.2 Testy stahování dat ČOI

Při stahování dat ČOI nebylo nutné testovat korektnost, jelikož prakticky všechna data se jen stáhla a neúčastnila se žádných transformací. Úplnost se testovala porovnáním počtů hlavních atributů (z každé datové sady ČOI alespoň jeden) v datovodu *EC test uplnosti*.

4.3 Testy integrace dat SVS a ČOI

Korektnost transformací, které při integraci dat probíhají testuje sada datovodů *TIN SVS to APP*, *TIN COI to APP* a *TIN APP to APP*. Testy probíhají na bázi příkladový vstup, očekávaný výstup. Umělé vstupy se tvoří uvnitř datovodů a testování, zda výsledky transformace jsou správné, probíhá pomocí PHPunit testů ve složce *web/tests*. Více detailů níže.

Zda se při integraci ztratila nebo neztratila konkrétní data, bylo možné testovat jen napůl. Bylo to zapříčiněné jednoduchou skutečností, že pokud ČOI i SVS

kontrolovala stejnou osobu, v integrované databázi by o ní byl pouze jeden záznam. To samé platí pro provozovny. Nakonec se také odstraňují duplikace. Nešlo tedy přesně říct, kolik osob a provozoven se mělo v integrované databázi očekávat. Ostatní šlo kontrolovat přesně. Testy integrace zajišťuje datovod *IN test uplnosti*. Z jeho výsledků lze vyčíst, co přesně se testovalo a jak. Ve výsledcích je uvedeno, zda počet konkrétních dat přesáhl minimální požadovanou mez a také zda přesáhl předpokládanou odhadnutou mez. Původ mezí je ve výsledcích také slovně popsán. Text SPARQL dotazu, který výsledky počítá, je v daném datovodu a navíc také v příloze *testovani uplnosti integrace.rq*.

4.4 Testy webové aplikace

Testy navržených scénářů použití webové aplikace zajišťuje balíček PHPunit testů, který také testuje správné fungování práce se souřadnicemi a měření vzdáleností na mapě. Tyto testy zároveň ověřují korektnost integrace dat ČOI a SVS. Všechny spustitelné testy mají název ve tvaru **Test.php*, ostatní soubory ve stejné složce (*web/tests*) jsou pouze pomocné.

Před testováním webové aplikace je potřeba spustit datovody *TIN COI to APP* a *TIN SVS to APP*. Po nich se ještě musí spustit *TIN APP to APP* a v kódu webové aplikace ve třídě *GLOB* je potřeba nastavit přepínač *$TESTOVANI_DB* na *true*. Díky tomu se jako zdroj dat pro webovou aplikaci použije testovací graf.

4.5 Výsledky testů

Po poslední aktualizaci dat v integrované databázi dne 1.7.2016 byly spuštěné všechny výše zmíněné testy. Přesto že se nepodařilo vytěžit úplně všechna data SVS, výsledky testů jsou považované za kladné. Problémy s vytěžováním totiž byly způsobené vadnými daty na straně SVS, což se nedá nijak obejít. Chybných dat bylo ostře méně než půl procenta z celkového objemu dat SVS.

4.5.1 Výsledky testů ČOI

Testy úplnosti při stahování dat ČOI proběhly zcela úspěšně.

4.5.2 Výsledky testů SVS

Výsledky testů posledního běhu všech datovodů při stahování dat SVS byly následovné. U tabulky *EU 1* chyběl jeden detail z 2576, jinak bylo vše v pořádku. U tabulky *OO 5* chyběly dva detaily z 86, jinak bylo vše v pořádku. U tabulky *PP 1* chyběly 3 detaily z cca 3873, jinak bylo vše v pořádku. U tabulky *EU 4* chyběly rybníky jednoho z 372 produkčních podniků, jinak bylo vše v pořádku. U ostatních tabulek se stáhlo vše správně. Co se týká stahování kontrol z webové aplikace, všechny testy byly v pořádku až na pár výjimek. Na mapě *Kontroly v místech určení (komodity z dovozu)* nebyly 2 JSONy z 861 validní. Na mapě *Kontroly v masných výrobnách* nebyl 1 JSON z 663 validní. Na mapě *Kontroly v drůbežích porcovnách* se nestáhla 1 závadná kontrola. Pravděpodobně šlo

o chybu v datech SVS v JSONu s detailem bodu, JSON byl sice validní, ale nebyla v něm data. Na mapě *Kontroly v mrazírnách* se nestáhly 2 závadné kontroly. Pravděpodobně šlo o prázdný JSON. Na mapě *Kontroly v tržní síti* se nestáhly 4 závadné kontroly. Pravděpodobně šlo o prázdný JSON. U ostatních map bylo vše v pořádku. Všechny chyby byly velmi pravděpodobně zapříčiněné chybnými daty SVS.

4.5.3 Výsledky testů integrace

Integrace proběhla v pořádku. Všechny testy korektnosti integrace proběhly bez chyby. Co se týká testů úplnosti, počty konkrétních atributů v integrované databázi vždy přesáhly minimální požadovanou mez a ve většině případů překročily i nezávaznou odhadovanou mez.

4.5.4 Výsledky testů webové aplikace

Webová aplikace se testovala proti navrhovaným scénářům použití webové aplikace. Testy webové aplikace proběhly úspěšně.

5. Uživatelská dokumentace

Webová aplikace umožňuje hledání podnikatelů nebo provozoven, které patří nějakému podnikateli 5.1. Hledat lze podle názvu, IČ a oboru podnikání. Výsledkem hledání je seznam odpovídajících podnikatelů nebo provozoven a hodnocení jejich poctivosti 5.5. Pokud je výsledků více, zobrazí se seřazené od nejméně poctivého po nejpoctivějšího. Dále lze aktivovat rozšířené vyhledávání 5.2 a hledat specifičtěji podle data kontrol, pouze v konkrétním kraji nebo pouze v nějaké konkrétně zadané oblasti 5.3 5.4. Lze obrátit seřazení a řadit od nejpoctivějších k nejméně poctivým. O provozovnách a jejich kontrolách lze zjistit podrobnější informace sledováním odkazů 5.6.

Navíc v nabídce *top 30* lze nahlédnout třicet nejdražších sankcí a další zajímavosti. Co se týče chybových hlášení, neměla by nastat a pokud nastanou, jsou jednoduše pochopitelná (jediný problém, kdy by mohlo dojít k chybě, by bylo při přetížení nebo výpadku databáze).

Obrázek 5.1: Možnosti vyhledávání ve webové aplikaci

Obrázek 5.2: Rozšířené vyhledávání ve webové aplikaci

Obrázek 5.3: Vyhledávání na mapě ve webové aplikaci

Obrázek 5.4: Vyhledávání na mapě ve webové aplikaci, druhá část

Obrázek 5.5: Výsledky hledání provozoven ve webové aplikaci

Obrázek 5.6: Detaily kontrol provozoven ve webové aplikaci

Závěr

Pro práci jsme vybrali dva kontrolní orgány veřejné správy - Českou obchodní inspekci (ČOI) a Státní veterinární správu (SVS). Data ČOI byla otevřeně dostupná v podobě Linked Data ve formátu RDF. Pro data SVS jsme navrhli RDF ontologii, která využívá známých slovníků a také má mnoho společného s ontologií dat ČOI, čímž se data z obou zdrojů sjednotila. Data o kontrolách SVS jsme posléze vytěžili z jejich webové aplikace a převedli do podoby RDF do zmíněné ontologie. Data ČOI jsme relativně jednoduše stáhli.

Data ČOI a SVS jsme správně zkombinovali a tak vznikla integrovaná databáze. Nad ní jsme nakonec vytvořili webovou aplikaci, která k integrovaným datům dává přístup neprogramátorské veřejnosti. Aplikace poskytuje možnost hledat podnikatele a jejich provozovny, zjistit kolik u nich proběhlo kontrol, jaké závady byly nalezené atd. Přestože je aplikace plně funkční, slouží spíše jako ukázka toho, k čemu je možné integrovanou databázi využít. Většina objemu práce totiž spočívala ve vytěžování dat a jejich integraci. Pro vytěžování dat SVS, stahování dat ČOI a jejich společnou integraci jsme použili ETL nástroj UnifiedViews. Značnou část kódu tvoří SPARQL dotazy uvnitř jednotlivých DPU v datovodech UV (cca 100 až 200 řádků na jeden datovod).

5.1 Hlavní přínosy

Mezi hlavní přínosy této práce patří vytvoření datové sady záznamů o kontrolách SVS. Datová sada je ve tvaru Linked Data ve formátu RDF a je zatím (4.7.2016) dočasně dostupná pro programátory na adrese `http://student.opendata.cz/sparql` v grafu http://vypod.cz/svs (data byla naposledy aktualizovaná dne 1.7.2016). Data používají nově navrženou ontologii, která je sama o sobě dalším přínosem. Pokud by mělo v budoucnu dojít k nové implementaci vytěžování dat SVS, bylo by vhodné použít naši ontologii, protože používá známé slovníky a dobře se doplňuje s ontologií dat o záznamech kontrol ČOI, tak jak jsou publikované na doméně Opendata.cz. Dalším přínosem je sada datovodů nástroje UV, které lze v budoucnu použít pro aktualizaci dat SVS. V neposlední řadě je zde webová aplikace, která přináší funkcionalitu vyhledávání nad integrovanými daty ČOI i SVS. Takovou funkcionalitu aktuálně nenabízí žádná jiná aplikace.

5.2 Řešené problémy

Vytěžování dat SVS a jejich integrace s daty ČOI sice byla úspěšně naimplementovaná pomocí nástroje UV, ale práce s tímto nástrojem nebyla úplně ideální. Hlavní problém představovaly časté výpadky. Alternativou pro podobná díla by bylo použití novějšího nástroje Linked Pipes (viz Nečaský a kol., 2016), který by potenciálně mohl být lepší (a měl by být).

Vytěžování dat SVS nakonec bylo o dost těžší než se původně odhadovalo. Navíc se celá práce zkomplikovala, když bylo potřeba přepsat většinu datovodů, jelikož se změnila struktura stránky SVS ještě předtím, než bylo vytěžování doladěné. Tomuto riziku se obecně nedá nijak předejít.

Při práci na vytěžování dat SVS, vyvstal velký problém při stahování dat o závadách. Zjistilo se, že id skupin kontrol jedné provozovny (tzv. bodů) se nedeterministicky mění, a proto nešlo stáhnout všechny záznamy o nalezených závadách. Tehdy jsme zavedli hypotézu, že změny nastávají jakoukoli manipulací s databází (tzn. manipulace s nějakými daty měnila id i úplně jiných dat, se kterými se nemanipulovalo). Na základě této hypotézy jsme navrhli stahovat data SVS jen o víkendu nebo přes noc, kdy byla manipulace nepravděpodobná. Experimentálně jsme pak ověřili, že v těchto dobách se id opravdu nemění. Není zde samozřejmě žádná garance, ale to ani není nutné. Kdyby náhodou došlo ke změně a nestáhla by se část dat, šlo by to vyčíst z výsledků datovodu a spustit jej znovu později.

5.3 Možnosti rozšíření

Webová aplikace nad integrovanou databází byla navržená tak, aby šlo jednoduše přidat nový datový zdroj. Vhodný kandidát na přidání je Český telekomunikační úřad. Je sice tématicky vzdálenější, ale data která publikuje mají podobný konceptuální model jako data ČOI a SVS. Takže by zdánlivě neměl být velký problém je vytěžit a přidat do integrované databáze.

Do budoucna je také potřeba počítat se změnou struktury webové aplikace SVS, změnou struktury publikovaných data a změnou rozsahu publikovaných dat. Jako rozšíření by tedy automaticky připadla v úvahu reakce na jakoukoli takovouto změnu.

Ke konci této práce ČOI přestala poskytovat otevřená data ve formátu RDF. Jediné co stále poskytovala byly exporty z databáze v CSV. Tím pádem zmizel zdroj aktualizovaných dat ČOI ve formátu RDF. Do budoucna by bylo vhodné vytvořit datovody pro převod dat ČOI z CSV do RDF. Toto by bylo zajímavé i z toho důvodu, že by u dat o kontrolách ČOI přibyl i obor podnikání provozovny (zakódovaný jako NACE). V dosavadních datech ČOI publikovaných ve formátu RDF totiž NACE z nějakého důvodu chybělo, i když v exportech z databáze uvedené bylo. K tomu by se vázal další cíl a tím je stažení překladů číselných kódů NACE do českého slovního popisu (viz Administrativní registr ekonomických subjektů, 2016) a obohacení dat ČOI.

Jako další rozšíření připadá v úvahu obohacení dat, u kterých chybí nějaká část adresy o tyto chybějící data. Speciálně nás zajímá kraj, kde se provozovna nebo osoba nachází. Tyto informace by měly být možné odvodit z PSČ.

Pro zlepšení webové aplikace by bylo vhodné přidat fulltextové vyhledávání podnikatelů a provozoven podle názvu (viz OpenLink Software, 2016a). K tomu by ovšem bylo potřeba spolupracovat s administrátorem serveru Virtuoso.

5.4 Obsah integrované databáze

Po poslední aktualizaci integrované databáze (dne 6.7.2016) v ní byla obsažena následující data: 29104 provozoven (z toho jen 1997 mělo uvedený název a obor podnikání a celých 27126 mělo uvedený kraj), 53768 kontrol (z toho 10446 závadných), 16526 podnikatelů a 47 různých oborů podnikání.

Přičemž každá provozovna měla uvedeného majitele a alespoň jednu kontrolu. Majitel měl uvedené IČ. Každá provozovna měla také uvedené souřadnice, kde se nachází. Každá kontrola měla uvedené datum a provozovnu, kde se konala. Každá závada (ať už ji zjistilo ČOI nebo SVS) měla také uvedenou kontrolu, při které byla zjištěná. Celkem se jednalo o necelý milion RDF trojic v integrované databázi.

Seznam použité literatury

Administrativní registr ekonomických subjektů (2016). Seznam cz-nace. *http://wwwinfo.mfcr.cz/ares/nace/ares_nace.html.cz (online).*

Beckett, D., Berners-Lee, T., Prud'hommeaux, E. a Carothers, G. (2014). RDF 1.1 turtle. *W3C Recommendation.*

Charles University in Prague (2016). Otevřená datová infrastruktura. *http://opendata.cz/ (online).*

Charles University in Prague a EEA (2016). open-source softwarový ETL nástroj vyvinutý na MFF UK. *http://unifiedviews.eu (online).*

Cibulka, J. (2015). Mapa: Které pumpy dostaly loni pokutu za nekvalitní palivo? *rozhlas.cz/brno.*

Česká obchodní inspekce (2016). Působnost úřadu. *http://www.coi.cz/cz/o-coi/pusobnosturadu/ (online).*

Český telekomunikační úřad (2016). Jaké spory ČTÚ řeší a kdy se na něj můžete obrátit. *https://www.ctu.cz/resene-spory-ctu (online).*

Google (2016a). Web services, geocoding api. *https://developers.google.com/maps/documentation/geocoding/intro (online).*

Google (2016b). Map coordinates, google maps api documentation. *https://developers.google.com/maps/documentation/javascript/maptypes (online).*

Harris, S., Seaborne, A., Prud'hommeaux, E., Garlik, Apache software foundation a Team Exparian (2013). SPARQL 1.1 Query Language. *W3C Recommendation.*

Linked Data community a Heath, T. (2016). Linked data — linked data - connect distributed data across the web. *http://linkeddata.org/ (online).*

Nečaský, M., Klímek, J. a Škoda, P. (2016). LinkedPipes, suite for Linked Data. *http://etl.linkedpipes.com/ (online).*

Nejvyšší kontrolní úřad (2016). Úvodní stránka. *http://nku.cz/default.htm (online).*

Open Knowledge Foundation (2016). Linked open vocabularies. *http://lov.okfn.org/dataset/lov/terms (online).*

OpenLink Software (2016a). Using full text search in SPARQL, Virtuoso documentation. *http://docs.openlinksw.com/virtuoso/sparqlextensions.html (online).*

OpenLink Software (2016b). Virtuoso functions guide. *http://docs.openlinksw.com/virtuoso/fn_st_point.html (online).*

SCHEMA.ORG COMMUNITY (2016). Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the internet, on web pages, in email messages, and beyond. *http://schema.org (online).*

STÁTNÍ VETERINÁRNÍ SPRÁVA (2016a). O státní veterinární správě. *http://eagri.cz/public/web/svs/portal/zakladni-informace/ (online).*

STÁTNÍ VETERINÁRNÍ SPRÁVA (2016b). Mapové výstupy z kontrolní činnosti státní veterinární správy. *http://eagri.cz/public/app/svs_pub/mapy_vk/ (online).*

STÁTNÍ VETERINÁRNÍ SPRÁVA (2016c). Registrované subjekty svs. *http://eagri.cz/public/web/svs/portal/registrovane-subjekty/ (online).*

STÁTNÍ ZEMĚDĚLSKÁ A POTRAVINÁŘSKÁ INSPEKCE (2015). Úvod, kontrolní činnost SZPI. *http://www.szpi.gov.cz/ (online).*

TENGSHE, D. (2013). EPSG 3857 or 4326 for GoogleMaps, OpenStreetMap and Leaflet. *http://gis.stackexchange.com/questions/48949/epsg-3857-or-4326-for-googlemaps-openstreetmap-and-leaflet (online).*

W3C (2014). RDF current status. *https://www.w3.org/standards/techs/rdf (online).*

WILLIAMS, H. a KLÍMEK, J. (2014). Virtuoso 22023 error sr. *https://github.com/openlink/virtuoso-opensource/issues/119 (online).*

WWW.SPARQL.ORG (2016). SPARQLer query validator. *http://www.sparql.org/query-validator.html (online).*

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

AJAJ Asynchronus Javascript and JSON; základ moderních webových aplikací. Jedná se o technologii neblokujícího posílání malých kousků dat z webového serveru na klienta, a jejich následné zpracování Javascriptem, aniž by webový klient musel znovu načítat stránku.

APP Aplikační model; zkratka pro logický datový model integrované databáze

ČOI Česká obchodní inspekce; kontrolní orgán české státní správy (Ministerstvo průmyslu a obchodu)

DOM Document Object Model; výsledek parsování HTML stránky, strom s atributy

DPU Data Processing Unit; základní stavební jednotka ETL nástroje

ETL Extract, transform, load; ETL nástroj je silný nástroj na extrakci dat, jejich transformaci do RDF a nahrání na server

JSON Javascript Object Notation; jednoduchý formát pro serializaci dat a přenos po síti

NACE CZ-NACE; číselné kódy ekonomických činností v ČR

RDF Resource Description Framework; datový formát pro sémantický web popisující orientovaný graf

SPARQL SPARQL Protocol and RDF Query Language; dotazovací jazyk nad RDF databází

SRID Spatial Reference System Identifier; standardní identifikátor souřadného systému

SVS Státní veterinární správa; kontrolní orgán české státní správy (Ministerstvo zemědělství)

TTL Turtle - Terse RDF Triple Language; typ serializace RDF (viz Beckett a kol., 2014)

UML Unified Modeling Language; grafický jazyk, standardní způsob kreslení diagramů

UV Unified Views; starší ETL nástroj

WKT Well-known text; standard pro textový zápis vektorové geometrie

XSD XML Schema Definition; jazyk popisující strukturu XML dokumentu, důležité je, že pro serializaci dobře definuje základní datový typy jako jsou xsd:integer a xsd:string

---

CSS Cascading style sheets

HTML Hypertext markup language

JS Javascript

SQL Structured query language

URI Uniform resource identifier

XML Extensible Markup Language

Přílohy

5.4.1 Datovody

Jména datovodů podléhají jednoduché konvenci. Datovody pro extrakci dat SVS začsínají na *ES* (extrakce SVS), datovody pro stažení dat ČOI začínají na *EC* (extrakce ČOI), datovody na integraci dat a tvorbu integrované databáze začínají na *IN* (integrace) a nakonec datovody na testování integrace začínají na *TIN* (testování integrace). Datovody se nacházejí ve složce dist/datovody/.

ES EU 1.zip
ES EU 2.zip
ES EU 4.zip
ES EU 10.zip
ES PP 1.zip
ES OO 5.zip
ES mapa.zip
ES Testovani korektnosti.zip
ES vycisti graf svs.zip

EC.zip
EC test uplnosti.zip
EC vycisti graf coi.zip

IN COI to APP.zip
IN SVS to APP.zip
IN APP to APP.zip
IN test uplnosti.zip
IN vycisti graf app.zip
IN vycisti graf app−nespojene.zip

TIN COI to APP.zip
TIN SVS to APP.zip
TIN APP to APP.zip
TIN vycisti graf test−app.zip

5.4.2 Webová aplikace

Webová aplikace je mimochodem také (dočasně) dostupná online na adrese `http://vypod.funsite.cz/`.

dist/web − zdrojove soubory
dist/web/src − zdrojove soubory
dist/web/spql − SPARQL dotazy
dist/web/lib − externi knihovny
dist/web/img − obrazky
dist/web/tests − testy webu a zaroven integrace

5.4.3 Data a některé dotazy pro testování integrace

dist/test/coi.ttl
dist/test/svs.ttl
dist/test/testovani uplnosti integrace.rq

5.4.4 Ostatní

dist/ukazky/priklad dat v ontologii SVS.ttl
dist/ukazky/svs kody obdobi.json
dist/ukazky/svs nazvy map.json
dist/ukazky/svs seznam provozoven v konkretnim obdobi na konkretni mape.json
dist/ukazky/svs detail konkretni provozovny v konkretnim obdoni na konkretni mape.json
dist/ukazky/svs priklad detailu konkretni provozovny z registracni tabulky.html
dist/ukazky/svs priklad registracni tabulky.html
dist/ukazky/reprodukce chyby concat ceskeho literalu.txt