Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Importér verejných datasetov

Dokumentácia inžinierskeho diela

Tímový projekt

Vedúci tímu: Ing. Jakub Šimko Phd.
Ing. Marek Šurek

Členovia tímu: Bc. Matúš Brandajský
Bc. Lukáš Belaj
Bc. Tomáš Božik
Bc. Michal Hrutka
Bc. Tatiana Šlesariková
Bc. Janka Fabušová
Bc. Gabriel Csollei

Študijný program: ISS
November 2017

Obsah

Zoznam obrázkov:

1. Úvod

Obsahom tejto dokumentácie je pohľad na technickú stránku projektu vypracovaného v rámci predmetu Tímový projekt na tému Importér verejných datasetov.

Cieľom projektu je vytvorenie webového nástroja, ktorý nie-technickému používateľovi (napr. úradníkovi) umožní jednoduchý import datasetu určeného na zverejnenie na data.gov.sk. Nástroj na vstupe dostane dataset vo formáte s neznámou sémantikou. Úlohou nástroja je dataset transformovať tak, aby zodpovedal centrálne platnej schéme, na základe ktorej sa potom jednotlivé zverejnené datasety môžu prepájať.

Na tvorbe tohto projektu sa podieľajú Fakulta informatiky a informačných technológií, Úrad podpredsedu vlády SR pre investície a informatizáciu a softvérová firma xIT s.r.o.

2. Globálne ciele projektu

2.1 Pre zimný semester

V rámci zimného semestra sme si s produktovým majiteľom vytýčili tieto ciele, ktoré by sme chceli stihnúť. Sú to:

- **Navrhnúť/implementovať používateľské rozhranie** – Tu je potrebné používateľské rozhranie upraviť v čo najprijateľnejšom rozhraní. Úradníčka musí vedieť ovládať rozhranie.
- **Navrhnúť spôsob ako zo vstupu (testový dokument) vytvoriť triplety** – Väčšina formátov je uložená práve v exceli alebo v csv súbore a preto je potrebné, aby sme sa zamerali na jednotnú podobu spracovania dokumentov. Na takúto operáciu nám slúži práve OWL schéma, do ktorej sa dáta budeme snažiť spracovávať
- **Stiahnutie centrálneho modelu a jeho spracovanie pre porovnávanie -** na to aby bola aplikácia funkčná je tento krok dôležitý. Centrálny model spracujeme a uložíme do prijateľnej formy.
- **Pokročilejšie metódy spracovania -** Pokúsime sa navrhnúť pokročilejšie metódy spracovania atribútov zo vstupných súborov. Samozrejme, tu bude potrebné prečítať a spracovať veľa výskumného textu, aby sme správne pochopili problém.

2.2 Pre letný semester

V rámci letného semestra sme si s produktovým majiteľom vytýčili tieto ciele, ktoré by sme chceli stihnúť. Sú to:

- **Vytvorenie vyhľadávača –** Pokúsime sa vytvoriť vyhľadávač v prípade, že používateľ nenájde taký atribút, ktorý mu ponúkame. On si bude môcť daný atribút vyhľadať
- **Vytvorenie nového atribútu -** Ak nenájde používate nový atribút ani vo vyhľadávaní, má možnosť pridať nový atribút. Tieto atribúty nebudú zahrnuté priamo do CM, ale budú čakať na schválenie kompetentnou osobou

- **Vytvorenie pokročilých metód spracovania údajov –** Po príprave zo zimného semestra, sa posnažíme popracovať na pokročilejšom určení atribútov.
- **Načrtnutie mapovanie dát na CM –** Toto patrí medzi dôležité úlohy. Nebude to ľahké sa s týmto problémom vysporiadať. Je potrebne vytvoriť z neštrukturovaných dát, štrukturované dáta pomocou Centrálneho modelu. Každý importovaný dataset by mal mať vlastné previazanie medzi atribútmi.
- **Prispôsobenie UX používateľovi –** Budeme pracovať aj na UX dizajne a prispôsobiť obrazovky netechnickému používateľovi.

3. Analýza

Pretože sme sa s týmto problémom ešte nikdy nestretli alebo sme sa s ním stretli len okrajovo, bolo potrebné spraviť aj kus analýzy, aby sme mohli problém implementovať.

Prvým problémom je samotné sémantické spracovanie dát. Tu sme začali pracovať na našej analýze v tom smere ako nám odporučili naši vedúci projektu. Počas tejto analýzy sme sa stretli s pojmami ako RDF[1], OWL[2] alebo triplety. Preštudovali sme si aj existujúce riešenia sémantického spracovania dát, ako napríklad GraphDB[3] alebo VOWL[4].

Po preštudovaní problematiky bolo potrebné analyzovať konkrétnu problematiku, s ktorou budeme pracovať. Bolo potrebné naštudovať slovenské štandardy, čo je to centrálny model a ako s ním pracovať. Tu nám pomohol zdroj od vedúceho, kde boli všetky potrebné informácie k centrálnemu modelu[5]. Následne bolo potrebné pozrieť sa na dáta, s ktorými budeme pracovať[6]. Tu sa nachádzajú všetky vstupné datasety, ktoré je možné spracovať. Samozrejme sme sa pozreli ako to funguje v iných krajinách a na akej úrovni majú datasety iné krajiny, konkrétne Spojené kráľovstvo[7].

3.1 Research pokročilejších metód spracovania názvu stĺpcov

Počas zimného semestra sme si zadefinovali smery, ktorými sa budeme uberať v letnom semestri pri určovaní názvu stĺpcov. Medzi ne patri viacero kategórií:

- Named Entity recognition
- Skratky
- Predikcia názvu stĺpcu z obsahu
- Ontology population
- Rozdelovanie stĺpcov

---

[1] https://www.w3.org/RDF/

[2] . https://www.w3.org/OWL/

[3] https://hub.docker.com/r/ontotext/graphdb/
[4] http://vowl.visualdataweb.org/

[5] https://wiki.finance.gov.sk/pages/viewpage.action?pageId=16416782
[6] data.gov.sk

[7] data.gov.uk

- Synonymický slovník a podobnosť textov
- Reprezentácia riadkov

3.1.1 Named Entity recognition

Named entity recognition je podtriedou extrakcie informácií, ktorá sa snaží nájsť a klasifikovať menované entity v texte do vopred definovaných kategórií. V našom projekte sa bude zaoberať Named Entity recognition, práve predikovaním názvov stĺpcov pomocou riadkov, ktoré budú pod daným atribútom v stĺpci. Podľa toho je možné identifikovať viacero druhov názvov stĺpca. Celkovo v literatúre sa nevenovali nášmu problému a určovali atribúty ako sú mená, organizácie a lokácie. My si musíme tieto atribúty rozdeliť do podstatne viac tried, čo výrazne sťažuje riešenie problému. Pár som uviedol v ukážke.

Identifikácia viacerých druhov názvov stĺpca:

- Mená ľudí – krstné meno, priezvisko
- tituly – pred, za menom
- časové výrazy – dátum, rok, mesiac, deň
- penažné hodnoty
- percentá
- organizácie
- lokácie – mestá, okresy, kraje, štáty

Viacero možných riešení + ich kombinácia:

- stochatické systémy – HMM, naivný bayes
- systémy založené na zemepisných slovníkoch
- machine learning – LSTM, CNN, ML-CNN
- kombinácia HMM a CRF – pravdepodobnostný grafový model

Celkovo je potrebné v tomto probléme uvažovať o viacerých výzvach, ktoré bude potrebné vyriešiť:

Medzi prvý problém patrí reprezentácia textu a rozdeľovanie textov. Menované entity, často nie sú jednoduchými slovami, ale sú práve zloženými slovami ako napríklad: Národná banka Slovenska alebo Slovenská technická univerzita. Preto je potrebný nejaký predikčný model, ktorý zistí, či skupina tokenov patrí do tej istej entity. V našom prípade to taký problém nie je, pretože vo väčšine prípadov sa bude nachádzať v jednej bunke jedna pomenovaná entita. Vhodný je napríklad Viterbiho algoritmus[1] alebo HMM [2]

Ďalším problémom, ktorý môže nastať je rozlíšenie entít od seba. Je to vlastne schopnosť určenia klasifikácie menovanej entity, najmä v miestach, kde vzniká nejednoznačnosť. Napríklad taký Washington môže byť aj priezvisko aj mesto. Na takéto problémy sa používajú algoritmy ako HMM alebo iné štatistické modely.[3]

Nelineárne modely závislostí sa vzťahujú na schopnosť identifikovať viaceré tokeny, ktoré sú rovnaké. Napríklad MATUS, Matúš, MatúŠ. Alebo skratky ako STU a Slovenská technická univerzita. To by v našom prípade nemusel byť problém, pretože nepotrebujeme, aby sme jasne spojili, že STU a Slovenská technická univerzita je tá istá pomenovaná osoba, ale že spadajú do tej istej kategórie Organizácia.

Celkovým problémom v našej práci je určiť, ktorá pomenované entita bude spadať do ktorého atribútu. Napríklad príbuzná osoba a priezvisko. Či sa jedná o meno alebo formátované meno(Reálne atribúty v CM)[8] [9]

Trénovanie a testovanie:

V našom prípade, by bolo potrebné spraviť viacero trénovacích dát, kde by boli atribúty, ktoré by presne vystihovali daný stĺpec. Samozrejme na vytvorenie takejto vzorky by bolo potrebné veľa úsilia a ľudskej manuálnej práce, možno aj od samotných úradníčok. Testovanie by sa robilo pomocou predikcie oštítkovaného stĺpca. Pri nasadení problému do produkcie by bolo možné náš model trénovať počas určovania atribútov úradníčkami.

---

[8] http://www.datacommunitydc.org/blog/2013/04/a-survey-of-stochastic-and-gazetteer-based-approaches-for-named-entity-recognition

[9] http://nlp.town/blog/ner-and-the-road-to-deep-learning/

Existujúce riešenia:

Stanford Named Entity Recognizer (NER) – implementované aj v Jave, ale nenašiel som github kód k tomu. Model je možné natrénovať na 7 artibútov: `Location`, `Person`, `Organization`, `Money`, `Percent`, `Date`, `Time`. Zdrojáky sa dajú stiahnuť, ale nič moc. Neviem ako by sa to správalo na slovenských vzorkách.[10]

Najlepšie by bolo spraviť si modifikovaný NER, z nejakého existujúceho riešenia, kde by sme si určili všetky potrebné triedy v CM.

Možné smery:

- Text Classification
- Data Extraction
- Computational Linguistics - Výpočtová lingvistika zahŕňa skúmanie spôsobov, pomocou ktorých stroj zaobchádza s prirodzeným jazykom.[11]

3.1.2 Podobnosť textov

Porovnávanie textov je v súčasnosti veľmi častý problém. V našom prípade potrebuje porovnávať slová, vety alebo skratky. Jednou s najjednoduchších metód je porovnávanie slova písmeno po písmene. Ako výsledok porovánania týmto spôsobom môže byť počet znakov v ktorých sa môže string líši. Táto metóda sa nazíva Hammingova vzdialenosť. Ďalšou metódou je Levenshteinova vzdialenosť. Tento algoritmus funguje na princípe dynamického programovanie ktoré je veľmi podobné vyhľadávanie podreťazcov Na základe Hammingovej vzdialenosti. Na rozdiel od hammingovej vzdialenosti nehovorí o počte nezhodných znakov ale o minimálnom počte operácie ktoré treba vykonať aby boli reťazce rovnaké. Problém môže nastať že výsledok je vždy celé číslo. Existujú aj pokročilejšie algoritmy ktoré vrátia hodnotu v rozsahu od 0 do 100 podľa toho či sú reťazce identické alebo úplne odlišné. Výhoda takýchto algoritmov je že rozlišujú aj hodnotu o akú sa jednotlivé znaky líšia.

[10] https://nlp.stanford.edu/software/CRF-NER.shtml
[11] https://www.techopedia.com/definition/13825/named-entity-recognition-ner

Medzi najjednoduchší spôsob ako porovnať 2 rôzne texty je porovnať počet rovnakých tokenov. Token je slovo ktoré je normalizované čiže upravené podla našej potreby. V našom prípade je to slovo premenená na malé písmená a odstránená diakritika. Tento spôsob nám hovorí iba o prekrytí dvoch textov a preto nám v našom prípade veľmi nepomôže, pretože nám nepovie veľa o podobnosti textov a taktiež preto že naše texty sú extrémne krátke. Ďalšia o trochu presnejšia možnosť je Jaccardov koeficient. Je to koeficient ktorý vyráta počet rovnakých tokenov. Následne sa porovná počet rovnakých tokenov a vyráta sa ich prienik deleno ich zjednotenie

Prekrytie dvoch textov:

$$Overlap =' catatemouse' cap' mouseatecatfood' = 3$$

Jaccardov koeficient:

$$Jaccard = frac|tokens_i n_s tring_A captokens_i n_s tring_B||tokens_i n_s tring_A cuptokens_i n_s tring_B|$$

Zoznam algoritmov na vyhľadávanie textov

- Brute force
- Morris-Pratt
- Knuth-Morris-Pratt
- Colussi
- Galil-Giancarlo
- Reverse Colussi
- Apostolico-Crochemore
- Skip Search
- KMP Skip Search
- Boyer-Moore
- Tuned Boyer-Moore
- Turbo Boyer-Moore
- Apostolico-Giancarlo
- Horspool
- Quick Search
- Zhu-Takaoka
- Berry-Ravindran
- Smith
- Raita

- Deterministic Finite Automaton Search
- Simon
- Reverse Factor
- Turbo Reverse Factor
- Backward Oracle Matching
- Forward DAWG Matching
- Karp-Rabin
- Not So Naive
- Two Way

S vyššie menovaných algoritmov je každý špecifický pre istý druh textov. Algoritmus ktorý berie text malej dĺžky(v našom prípade najčastejšie použitý) čo je do 20 znakov a veľkosť abecedy do 100 znakov tak najlepšia voľba je Two Way, Not So Naive alebo Skip Search

3.1.3 Ontology population

Ontology population je automatické alebo polo-automatické vytváranie ontológií, vrátane extrahovania korešpondujúcich termínov z určitej domény a vzťahov medzi konceptami, ktoré tieto termíny reprezentujú z korpusu textov prirodzeného jazyka a zakódovaním ich do ontologického jazyka pre jednoduché spracovanie. Nakoľko je manuálne vytváranie ontológií veľmi náročné, aj čiastočná automatizácia výrazne urýchľuje tento proces.

Proces:

1. Extrakcia terminológie domény Relevantné termíny pre danú doménu možno získať pomocou kalkulácie tf-idf skóre. Výsledný zoznam musí byť prefiltrovaný doménovým expertom. Následne sú identifikované synonymá, keďže majú rovnaký význam. Najčastejšie používané metódy sú clustrovanie a miera štatistickej podobnosti

2. Odhalenie konceptu V tomto kroku sú termíny a ich synonymá identifikované v predošlom kroku zgrupované do jednotiek, ktoré korešpondujú s abstrakciou reálneho sveta a teda s konceptom.

3. Odvodenie hierarchie konceptu Koncepty sú usporiadané do taxonomickej štruktúry, čo je zvyčajne dosiahnuté metódami hierarchického clustrovania bez učiteľa. Nakoľko sú výsledky neveľmi presné, využíva sa v tomto kroku evaluácia používateľom.

4. Zistenie netaxonomických vzťahov Extrahujú sa vzťahy, ktoré nepredstavujú pod- alebo nad-triedy. Existujú dva bežné prístupy. Prvý je založený na extrakcii anonymných asociácií, druhý na extrakcií slovies, ktoré indikujú vzťahy medzi entitami, ktoré sú reprezentované okolitými slovami. Výsledky oboch prístupov musia byť vyhodnotené ontogológom.

5. Odhalenie pravidiel Axiómy – formálne opisy konceptov – sú generované pre extrahované koncepty tak, že sa analyzuje syntaktická štruktúra definície prirodzeného jazyka a aplikujú sa transformačné pravidlá na vzniknutý strom závislostí. Výsledkom je zoznam axiómov, ktoré sú zahrnuté do popisu konceptu. Výsledok musí byť vyhodnotený ontológom.

6. Populácia ontológie V tomto kroku je ontológia rozšírená o inštancie konceptov a vlastností.

7. Rozšírenie hierarchie konceptu Taxonomická štruktúra existujúcej ontológie je rozšírená o ďalšie koncepty. To môže byť dosiahnuté pomocou natrénovaného klasifikátora alebo aplikáciou metriky podobnosti.

8. Detekcia udalosti a rámca Extrakcia komplexných vzťahov z textu, napríklad kto vyrazil odkiaľ a kedy. Dosiahnuteľné pomocou SVM s kernelovými metódami.

Na určenie nejakého vzťahu medzi dvomi artefaktmi sú potrebné doménové fakty. Doménový fakt obsahuje element zo zdrojového artefaktu (napr. prístupové práva), element z cieľového artefaktu (napr. privilégiá) a typ vzťahu (napr. sú). Cieľom je využiť vzťahy medzi zdrojovým a cieľovým artefaktom za účelom generovania dvojíc, následne aplikovať webmining, NLP a techniky strojového učenia na dokázanie, že by daná dvojica mohla reprezentovať významný doménový fakt. Počet takýchto dvojíc rapídne rastie s narastajúcou dĺžkou artefaktov a väčšina dvojíc nereprezentuje validné doménové fakty a je potrebné vyfiltrovať irelevantné dvojice.

Na identifikáciu dvojíc sa používa Part-of-Speech tagger a extraktor fráz, použité primárne na extrakciu podstatných mien, fráz a slovies. Každá dvojica pozostáva z 1 termínu alebo frázy zo zdrojového artefaktu a 1 termínu alebo frázy z cieľového artefaktu. Následne sa zbierajú informácie z iných vedomostných zdrojov, ktoré zahŕňanú Lexico-syntactic pattern matching, association rule mining, topic modeling a semantic relatedness. Lexikálno syntaktické vzory reprezentujú rekurentné výrazy v prirodzenom jazyku.

LSP matching prehľadáva doménové dokumenty za účelom nájdenia priamych dôkazov na podporu správnosti dvojice. Napríklad, nájdenie frázy „používateľské privilégiá zahŕňaju prístupové práva“ je priamym dôkazom toho, že „prístupové práva“ sú podtriedou „privilégií“. Dolovanie asociačných pravidiel vypočíta ako často sa kombinácia termínových/frázových

párov vyskytuje naprieč všetkými spojeniami. Modelovanie topikov vypočíta do akého rozsahu termíny a frázy súvisia s rovnakým topikom Sémantická spojitosť opisuje sémantickú podobnosť termínov/fráz na základe externého zdroja ontológie. Získané informácie sú použité na natrénovanie Random Forest klasifikátora na manuálne vytvorenom trénovacom datasete.

3.2 Analýza datasetov z data.gov

Analyzovali sme aj datasety s data.gov.sk, hľadali sme najhoršie datasety a snažili sme sa určiť, ktoré metódy by boli vhodné na konkrétne datasety. Bolo analyzovaných 25 datasetov.

3.2.1 Analýza 1

1csv_statistika_podla_druhu_kriminality_31.10.2012.csv

- prvým problémom je, že riadky nezačínajú na prvom riadku, ale reálne to začína až na 7dmom riadku, takže tu bude problém so spracovaním.
- atribúty sa v centrálnom modeli vôbec nenachádzajú a bolo by dobré, aby sa do budúcna definovali
- V niektorých atribútoch vystupujú skratky ako t.j, alebo len skrátené slova ako napríklad Dodatoč. objasnené.

Vhodné metódy do budúcna: **Skratky, Predikovať názov atribútu z obsahu stĺpca –** prvý stĺpec

3csv_statistika_podla_paragrafov_31.10.2012.csv

- tento dataset je trochu podobný prvému.
- atribúty sa opakujú – duplikáty
- väčšina výrazov je písaná heslovite napr. Maloletý –drogy alebo Mladistvý – alkohol
- celkovo sú tam duplikáty kvôli tomu, že sa nachádzajú ako keby dve časti v jednom datasete, prvá časť o trestných činoch a druhá o vyšetrovaných osobách
- tiež tieto atribúty sa nenachádzajú v CM

Vhodné metódy: **synonymický slovník, skratky, možno predikcia podľa obsahu**- ak by som našiel paragraf dokázal by som sa dobre nasmerovať pri určovaní

08namietkyneplatnostzruseniefinal201711

- je tam atribút id, čo treba úplne vyhodiť
- pri atribúte číslo prihlášky mi š zobrazilo divne, tu bude treba dávať pozor na divné znaky
- zase sa atribúty nenachádzajú v CM

Vhodné metódy: Levensteinova vzdialenosť, synon. slovníky

ARCHIVNE_POMOCKY.csv

- atribúty sú úplne nečitatelné
- tu by bolo dobré nahradiť _ za medzeru trochu by to pomohlo.
- zase, niektoré „pofidérne“ skratky

Vhodné metódy: Skratky, Named entity recognition – na pár stĺpcov, tuto by bolo dobré asi aj osloviť úradníka, lebo to je moc.

crs.csv

- dataset nie je tradičný, prvý stĺpec vyjadruje kraje a ďalší vyjadruje vek mužov a žien a či pochádzajú zo Slovenska alebo nie, takže sú všetky podobné
- keby sme si odmysleli prvý riadok, tak by sa s tým dalo pracovať
- taktiež v CM sa nachádza len osoba, ale nie muž alebo žena

Vhodné metódy: ak budú údaje v CM, tak to bude dosť jasné.

Databáza základných škôl zapojených v NP PRINED).csv

- slová sú oddeľované _ nie medzerou
- dataset začína klasicky od prvého riadku
- s týmto by sa dalo pracovať keby sa zadefinujú všetky atribúty do CM
- niektoré atribúty ani netuším čo by mali byť napríklad HEI alebo SZP

Vhodné metódy: Skratky, Named entity recognition- tu by sa dalo viacero atribútov, podľa obsahu stĺpca tiež.

dovodyodmietnutia2016150.csv

- malý dataset, ale zato sa tu nachádza anglický názov, ktorý ani nie je oddelený medzerou CallCenterName
- ostatné názvy sú zase po slovensky
- pri tom CallCenterName tam by sa dalo použiť named Entity recognition, pretože sú tam mestá, ale či by to určilo akurát toto čo mysleli to neviem.

Vhodné metódy: Podobnosť textov, Named entity recognition, Predikovať názov atribútu z obsahu stĺpca

Export služieb IS.csv

- Každý atribút je síce po slovensky ale nie sú oddelené čiarkou napr., CharakterSluzby alebo FazaSluzby.
- Všetky atribúty sú takto napísané
- Tu by nám pomohli skratky na niektoré veci (IsvsNazov) ale najviac asi podobnosť slov.
- je tam aj verzia a v každom riadku potom 1.0, toto by nám mohlo pomôcť ho určiť ak by to nebolo jasné

Vhodné metódy: Podobnosť textov, skratky

fakulty.csv

- v tomto datasete by sa nám s pravdepodobnosťou podarilo určiť nejaké atribúty
- sú tam niektoré jasné ako PSČ alebo ulica
- sú tam atribúty ako názov a skratka kde by nám pomohli named entity recognition

Vhodné metódy: Named entity recognition, skratky, Predikovať názov atribútu z obsahu stĺpca

mvsrplanzverejnovaniadatasetovn.csv

- tak toto je celkom dosť. Tie atribúty majú priemer 8 slov asi
- najdlhší atribút: Zverejniteľnosť (napr. "už zverejnené", "možné zverejniť v celom rozsahu", "možné zverejniť len po anonymizácii / úprave", "možné zverejniť len v agregovanej podobe" a pod.)
- tuto nám pomôže rozdeliť to čo slovo to dopyt a následne pomocou synoným to hľadať

Vhodné metódy: Named entity recognition, Predikovať názov atribútu z obsahu stĺpca, synonymá, asi aj Rozdeľovanie stĺpcov

plneniefinancnehoplanu

- toto nie je celkom tradičný dataset
- začína reálne od 4tého riadku
- celkovo prvý stĺpec vyjadruje viac atribútov aj náklady a výnosy
- toto neviem ako presne by sme mali spracovať

Vhodné metódy: Rozdeľovanie stĺpcov, skratky

poetpristupovnaupvszade

- tento dataset mi príde úplne nezmyselný
- tu vôbec netuším o čo sa jedná a aký účel bol tohto datasetu

Vhodné metódy: Možno nejaké synonymá

poetrozhodnutirealizovanychcezu

- malý dataset, len 2 atribúty
- jeden by sa dal určiť a druhý nie je v CM, treba doplniť a pôjde to

Vhodné metódy: tu asi netreba žiadne, možno synonymá

poetzriadenychelektronickychsch

- podobný dataset ako predtým
- 3 atribúty
- tu treba asi nejaké synonymá, neviem celkom. Záleží od toho ako budú zadefinované atribúty v CM

Vhodné metódy: tu asi netreba žiadne, možno synonymá

prehladcentralnejevidenciehospodarskychzvieratipolrok2017.csv

- tento dataset je divný
- má atribúty v riadkoch a nie v stĺpcoch a nezačína ani prvom riadku
- v stĺpcoch má vyhradené obdobie a možno aj nejaký atribút napr. prírastok v %

Vhodné metódy: nejak zistiť že sú atribúty v riadkoch

stavintegracienaupvs

- polka datasetu ani nie je vyplnená
- tie ktoré sú tak tam by nám pomohlo Named entity recognition alebo Predikovať názov atribútu z obsahu stĺpca
- Určite by nám pomohli aj skratky: UAT podpis, PROD nasadenie, Rola ÚPVS

Vhodné metódy: skratky, Named entity recognition, Predikovať názov atribútu z obsahu stĺpca

tabulkaopensourcemvsr

- tuto zase dlhé atribúty na 10 slov a viac
- v každom atribúte na konci bodkočiarka, možné odstránenie pre lepšie dosiahnutie výsledku
- Rezort / Organizácia: - tu je to také že sa nevedia rozhodnúť čo to vlastne je

Vhodné metódy: skratky **– OSS,** Named entity recognition, Predikovať názov atribútu z obsahu stĺpca, asi aj synonymá.

ustrednekontaktnecentrum

- tu je jeden atribút dokopy: Vybavené hovory + kontaktný formulár celkom, ten by mal byť buď rozdelený na dva, ktoré sú tak nazvané predchádzajúce atribúty alebo spraviť z toho jeden atribút
- nezačína od začiatku

Vhodné metódy: synonymický slovník by dosť pomohol alebo Rozdeľovanie stĺpcov

Zoznam absolventov a záverečné prezentácie zo vzdelávania PZOZ.csv

- Tento je dosť jasný
- Tu by to asi pekne aj klasifikovalo
- tam kde je diakritika je otáznik, to neviem kde je problém.

Vhodné metódy: Named entity recognition, , Predikovať názov atribútu z obsahu stĺpca

Zoznam datasetov MŠVVaŠ SR

- takéto niečo tu už bolo
- dlhé názvy atribútov zase
- je to dosť podobné ako to predtým, čo bolo

Vhodné metódy: Named entity recognition, Predikovať názov atribútu z obsahu stĺpca, synonymá, asi aj Rozdeľovanie stĺpcov

zoznamevidovanychfariemvcehz

- atribúty bez diakritiky a spojené do jedného slova aj viacslovné
- zase Každé nové slovo začína veľkým písmenom: NazovFarmy
- nejaké skratky ako GIS X
- je tu aj okrej aj kraj, aj obec, ulica, takže jednoznačne named Entity recognition
- Obec sa duplikuje

Vhodné metódy: Named entity recognition, Predikovať názov atribútu z obsahu stĺpca, synonymá, skratky

zoznamintituciisozriadenouelektronickouschrankou

- celkom klasický oproti ostatným
- pár skratiek PSČ
- zase ulica mesto, IČO

Vhodné metódy: Named entity recognition, Predikovať názov atribútu z obsahu stĺpca, synonymá, skratky

zoznamovmsaktivovanouschrankounadorucovaniektoremajumaterskuidentitu

- tuto celkovo málo atribútov
- dá sa niečo zistiť pomocou skratiek
- celkovo podobne ako ostatne

Vhodné metódy: skratky, synonymá

Životné situácie a eGovernment služby vyšších územných celkov

- tuto by sa dali asi využiť niektoré atribúty z eGovernment entity

Vhodné metódy: synonymá

щseky a agendy verejnej spravy

- atribúty sú zase v jednom slove všetky.
- niektoré atribúty sa dajú zistiť pomocou named entity recognition: NazovRezortu, NazovAgendy

Vhodné metódy: synonymá, named entity recognition

3.2.2 Analýza 2

Problémy:

- nedodržaná štruktúra *csv* súboru – napr. súbor obsahuje „hlavicku“ [1,2,5,10,17,20]
- hodnota, ktorá obsahuje čiarku nie je v úvodzovkách[1]
- ? štruktúra csv ako tabuľky – aj riadky majú názvy [1]
- nesprávne zobrazenie špeciálnych znakov, napr. „Mladistvý páchate¾“[2,4,8,17,19,21,22]
    - niekedy záleží na toole, zobrazuje správne v Atome alebo v Exceli
- ? viacero datasetov v jednom súbore – stĺpce sú ale rovnaké pre všetky [2,22]
- skratky [4,5,10,18,19,21,23,24,25,27]
- súbor začína oddeľovačom – čiarkou alebo bodkočiarkou [5,19]
- viacslovné názvy stĺpcov oddelené „_“ [7]
- viacslovné názvy spojené *camelCase* [8,10,20,23]
- mix anglických a slovenských názvov stĺpcov [9]
- veľmi dlhé názvy stĺpcov [12,19,22]
- úplný bordel – imho nespracovateľné [13,17]
- rôzny počet názvov stĺpcov a samotných stĺpcov [14,16,18,24,25]
    - napr posledný názov stĺpca „Naposledy aktualizované:...“

Výskumné smery:

- skratky
- kontrola špeciálnych znakov
- detekcia zbytočných stĺpcov ako „Naposledy aktualizované“
- rozdeľovanie camel case názvov

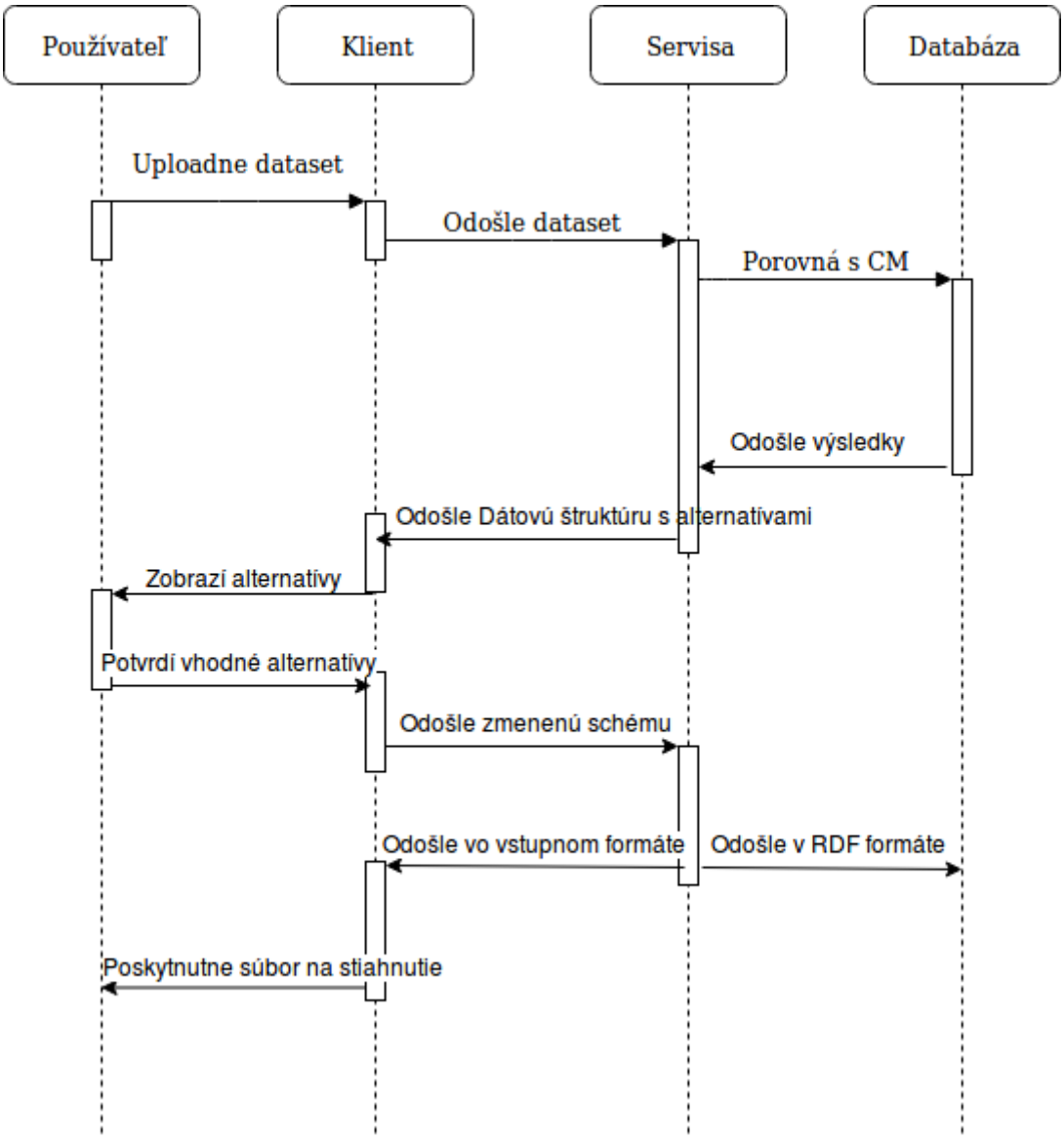
Súbory:

1. 08namietkyneplatnostzruseniefinal201711.csv
2. 1csv_statistika_podla_druhu_kriminality_31.10.2012.csv
3. 3csv_statistika_podla_paragrafov_31.10.2012.csv
4. ARCHIVNE_POMOCKY.csv
5. crs.csv
6. data.gov.sk.zip
7. Databáza základných škôl zapojených v NP PRINED).csv
8. dataset_analysis.docx
9. dovodyodmietnutia2016150.csv

10. Export služieb IS.csv
11. fakulty.csv
12. mvsrplanzverejnovaniadatasetovnarok2017.csv
13. plneniefinancnehoplanu.csv
14. poetpristupovnaupvszade.csv
15. poetrozhodnutirealizovanychcezupvs.csv
16. poetzriadenychelektronickychschranok.csv
17. prehladcentralnejevidenciehospodarskychzvieratipolrok2017.csv
18. stavintegracienaupvs.csv
19. tabulkaopensourcemvsr.csv
20. ustrednekontaktnecentrum.csv
21. Zoznam absolventov a záverečné prezentácie zo vzdelávania PZOZ.csv
22. Zoznam datasetov MŠVVaŠ SR.csv
23. zoznamevidovanychfariemvcehz.csv
24. zoznamintituciisozriadenouelektronickouschrankou.csv
25. zoznamovmsaktivovanouschrankounadorucovaniektoremajumaterskuidentitu.csv
26. Úseky a agendy verejnej správy.csv
27. Životné situácie a eGovernment služby vyšších územných celkov.csv

4. Návrh

Počas návrhu bolo nutné vytvorenie sekvenčného diagramu, ktorý nám ukazuje ako budeme postupovať a ako bude prebiehať chod našej aplikácie. Pomocou neho sme si vytvorili základný obraz o tom, ako bude naša aplikácia



Obrázok 1:Sekvenčný diagram: Celkový chod systému.

Na sekvenčnom diagrame je vidieť celý chod nášho systému:

1. V prvom kroku je nutné, aby používateľ systému(úradník) nahral dataset, ktorý chce spracovať.

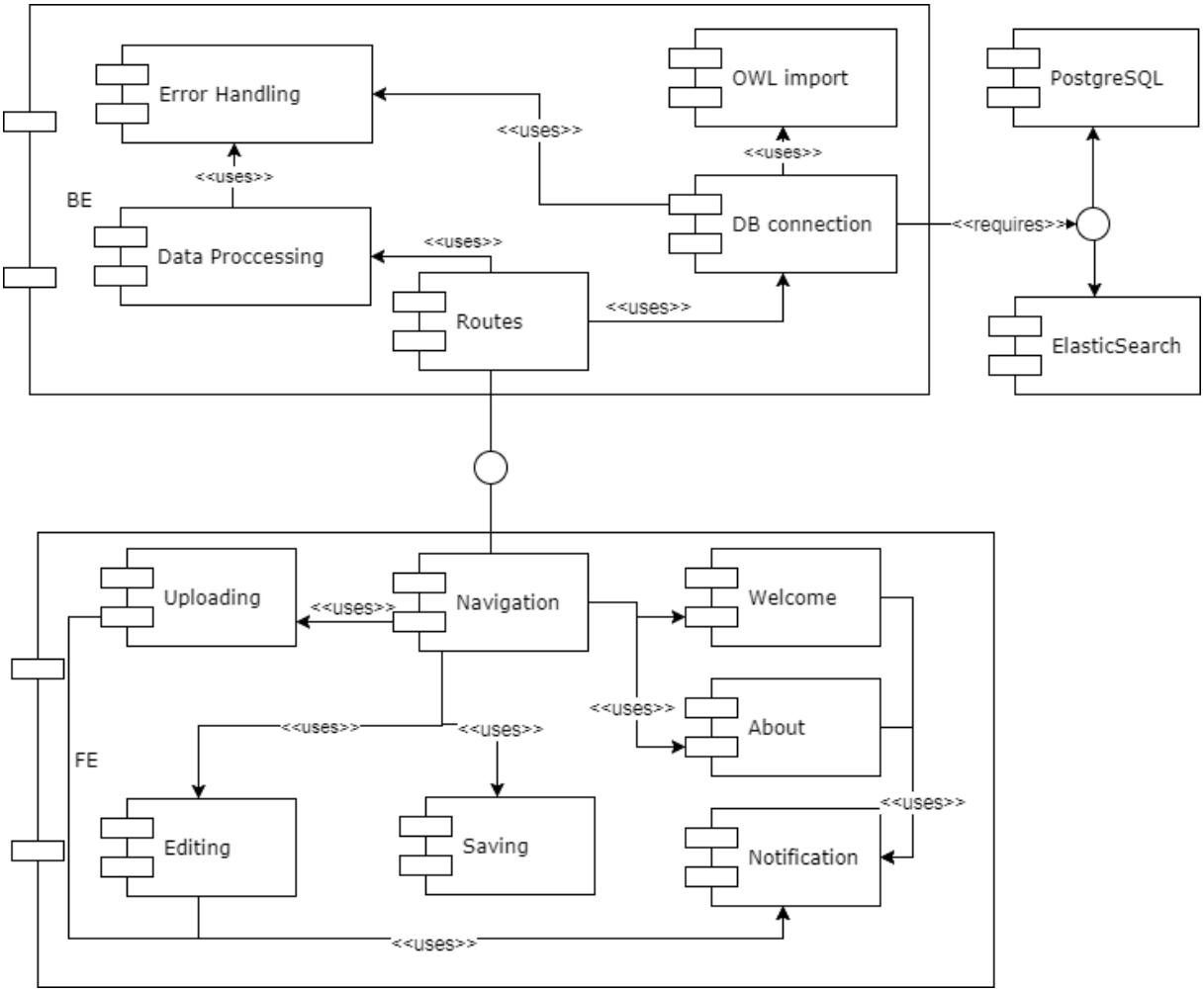
2. Následne sa dataset odošle na Backend, kde sa údaje spracujú s pomocou porovnávanie s centrálnym modelom. Tu sa zistia alternatívy, ktoré sa následne posielajú znovu na Frontend
3. Údaje sa zobrazia používateľovi na Frontende a on musí vybrať z každej alternatívy, ktorá mu bola poskytnutá pre daný atribút.
4. Po vybratí znovu Frontend odošle údaje na backend. Backend údaje spracuje do dvoch podôb. V prvej podobe odošle dataset na Frontend vo vstupnom formáte. Tu klient ponúkne používateľovi stihanuť si upravenú verzu svojho datasetu. V druhej podobe sa údaje uložia do RDF a následne sa uložia do našej databázy.

K nášmu projektu sme vytvorili aj klikateľný prototyp[12], ktorý korešponduje so sekvenčným diagramom. Tu je zapracovaný aj UX dizajn, na ktorom sme sa zhodli s našimi vedúcimi.

[12] https://janyfabusova.wixsite.com/team-14

5. Implementácia

5.1 Celkový pohľad na systém



Obrázok 2: Celkový pohľad na systém

Obrázok nad textom vyjadruje celkový pohľad na systém. Aplikácia je realizovaná pomocou klient - server architektúry. FE vyjadruje Frontend, teda klientskú časť, ktorú využíva používateľ a potom BE, čo je vlastne backend, teda serverovú časť. Klient komunikuje so serverom pomocou HTTP protokolu cez web rozhranie. Obrázok ďalej vyjadruje aj závislosti komponentov.

Identifikované komponenty spĺňajú nasledovné funkcie:

Navigation a Routes - Komponenty, ktoré slúžia na komunikáciu medzi klientom a serverom. Komponent Routes momentálne vyjadruje dve časti, prvou je prijatie nahratého súboru a druhé

je prijatie alternatív, po správnom vybratí dokumentu. Vo frontendovej časti je Navigation rozdelený do viacerých balíkov, ktoré sa taktiež starajú o presmerovávanie na stránke. Sem patrí napríklad objekt Router ktorý ponúka angular a jeho použitie je veľmi jednoduché. Volania backendových apín sú rozdelené do viacerých služieb, ktoré dedia od hlavnej AbstractHTTP Service. Ostatné frontendové komponenty sú popísané nižšie.

Data processing - Tento komponent slúži na spracovávanie nahratého datasetu. Rozpozná o aký dokument sa jedná a implementuje ho do java objektov.

Error handling - Tento komponent sa stará o chybové hlášky, ktoré máme pevne definové a je potrebné ich odoslať aj na Frontend. Pomocou nich sa následne používateľovi vypisujú chybové hlášky.

DB Connection - Komponent, ktorý zabezpečuje komunikáciu a pripojenie s Elasticom, zároveň robí aj všetky vyhľadávania nad elasticom.

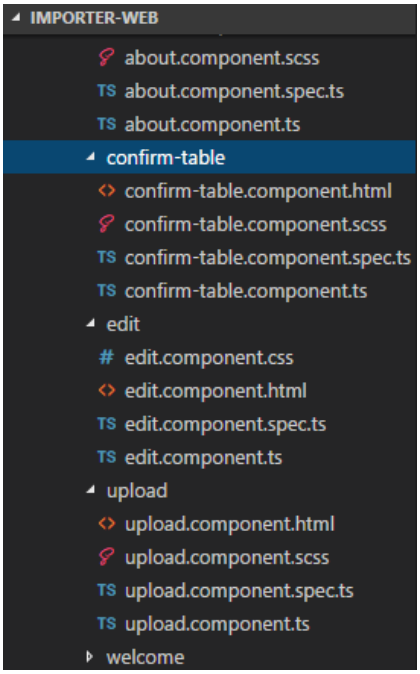
OWL Import - Komponent na spracovanie centrálneho modelu.

ElasticSearch - momentálne slúži na uchovávanie Centrálneho modelu a vyhľadávanie v ňom.

5.2 Frontend

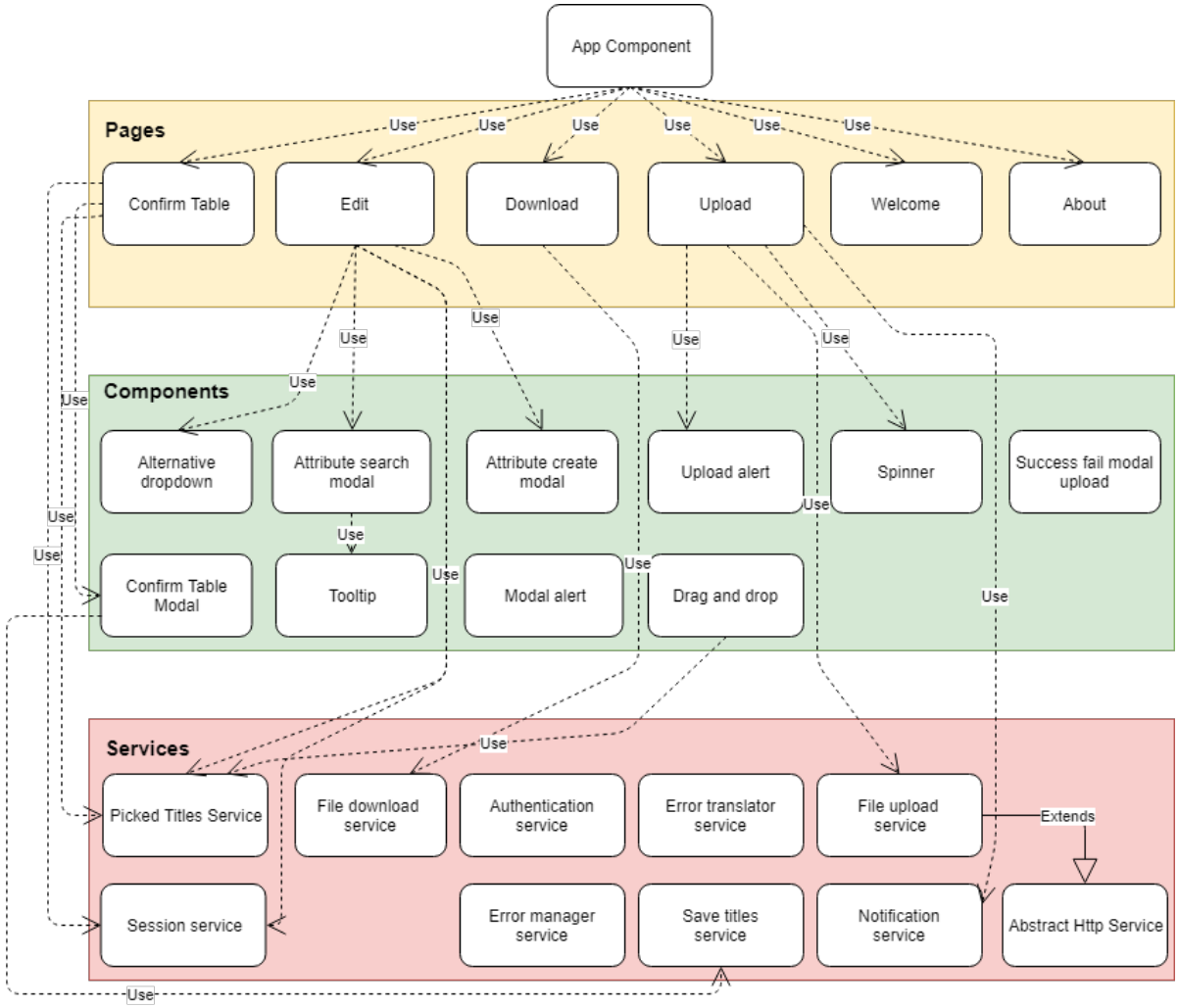
Frontendová časť aplikácie je postavená na modernej technológii Angular 4 od spoločnosti Google, ktorá komunikuje s API za pomoci REST-ových služieb. Tento rámec nám prináša množstvo výhod, na základe ktorých sme sa pre neho rozhodli. Okrem jeho rýchlosti, jednoduchosti a flexibilite má rôzne stratégie, ako tvoriť aplikácie pre rôzne platformy, či už pre webové prehliadače alebo mobilné zariadenia. Kód je členený na jednotlivé komponenty, čím sa stáva prehľadný a ľahko zrozumiteľný. Aplikácia je postavená na Angular-CLI, spolu s čím prichádzajú výhody spojené s inštalovaním závislostí, testovaním a budovaním aplikácie (angl. build application) pre produkčnú alebo vývojársku verziu.

Na obrázku Náhľad na štruktúrované priečinky podľa funkcie je možné vidieť, ako je zdrojový kód rozdelený na priečinky podľa funkcií, ktoré obsahujú komponent, HTML šablónu, službu, CSS štýly a používané triedy.



Obrázok 3: Náhľad na štruktúrované priečinky podľa funkcie

Na ďalšom obrázku (Diagram komponentov a služieb) môžeme vidieť základné služby a komponenty používané v našej aplikácií. Hlavný App komponent využíva všetky ostatné komponenty na základe aktuálnej URL. Upload service je služba, ktorá sa stará o upload súborov. Dedí od služby Abstract-http servive, ktorá poskytuje základné metódy na posielanie HTTP requestov. Ďalej tu je dôležitý Upload komponent, ktorý obsahuje komponent na výber súbor DragAndDrop component a taktiež aj Edit component, ktorý zabezpečuje výber všetkých atribútov. ConfirmTable component slúži na zobrazenie celkového prehľadu zvolených atribútov a ich následné odoslanie na data.gov, poprípade stiahnutie finálne upraveného dokumentu. Všetky komponenty využívajú službu na notifikáciu používateľa Notification service, ktorá spracuje prijaté chybové hlášky to server, umožňuje lokalizáciu aplikácie do viacerých jazykov a taktiež sa stará o jednoduché logovanie do konzoly.



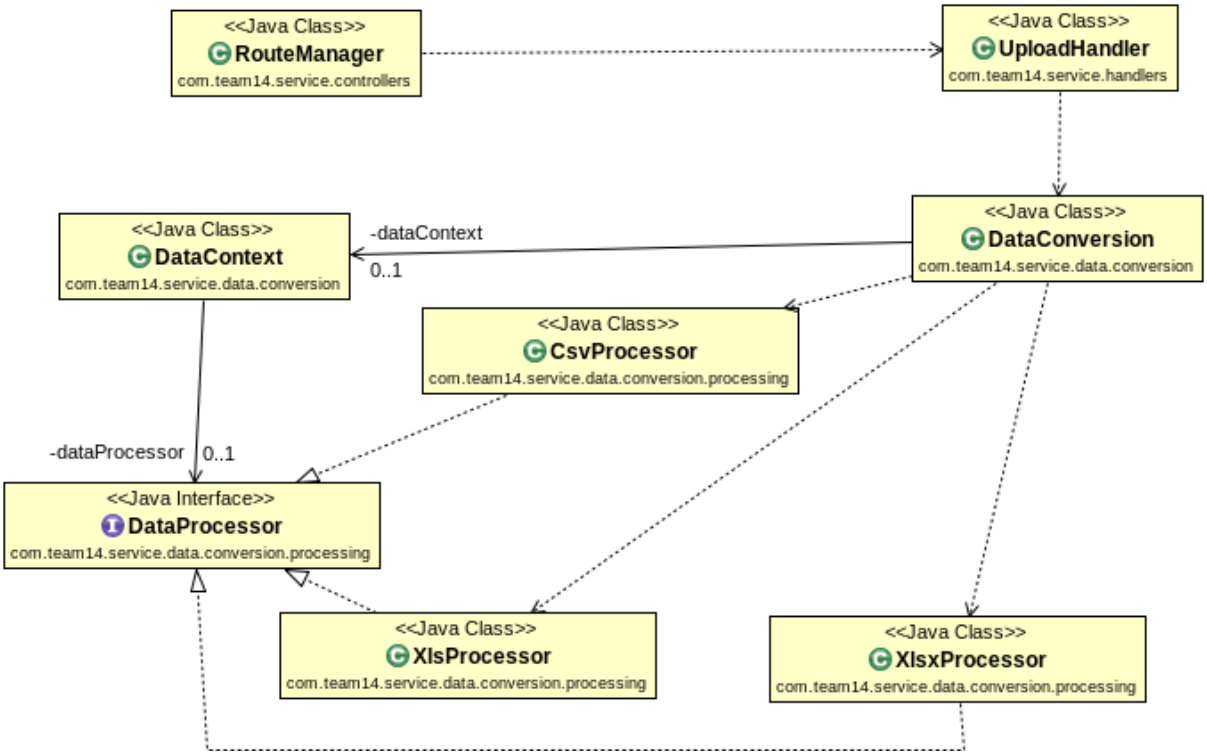
Obrázok 4: Diagram komponentov a služieb

Frontend našej aplikácie využíva technológiu Angular 4. Základom aplikácie je AppComponent, ktorý slúži ako master page a zahŕňa všetky ostatné stránky (Pages). Stránky sa skladajú z komponentov, ako napríklad dropdown alebo modálne okná. Samotné komponenty môžu byť ešte rozdelené na menšie komponenty ako napríklad tooltip, ktorý je súčasťou modálneho okna, pri čom modálne okno je súčasťou stránky. Okrem toho sa na komunikáciu s backendom a medzi komponentami využívajú služby (services). V aplikácii využívame aj rozhranie pre zjednotenie tvorby notifikácii a model pre dáta komunikované cez sieť, tie však pre sprehľadnenie nie sú v nákrese architektúry z rovnakého dôvodu chýbajú aj vzájomné závislosti medzi komponentami.

5.3 Backend

Backend je postavený na programovacom jazyku JAVA. Tento jazyk nám prináša mnoho výhod a preto sme sa preň rozhodli. Jednou z výhod je škálovateľnosť aplikácie. Zvolili

sme si framework Spring, ktorý nám uľahčuje celkovú prácu a otvára nám možnosti využitia aplikácie. Na komunikáciu používame REST služby. Aplikácia využíva buildovací framework maven, ktorý nám uhahčuje prácu a dovoluje nám integráciu s Dockerom, ktorý aktívne využívame. Dole na obrázku je vidieť celkový model našej aplikácie.



Obrázok 5: Spracovanie datasetu

RouteManager – Obsahuje všetky endpointy, ktoré môžu byť zavolané frontendom. Každý endpoint má vlastný handler, ktorý obsahuje funkcionalitu špecifickú pre danú úlohu.

UploadHandler – Táto trieda po odoslaní súboru na spracovanie tento súbor prečíta, spracuje, uloží na server a následne vyhľadá a vráti všetky alternatívne názvy stĺpcov pre všetky stĺpce súboru.

DataConversion – V tejto triede sa najskôr zistí typ uploadnutého súboru a následne sa tento súbor z byte-ového poľa presype do vnútornej reprezentácie (List Stringov) a zvlášť sa uloží prvý riadok zo súboru ktorý predstavuje názvy stĺpcov.

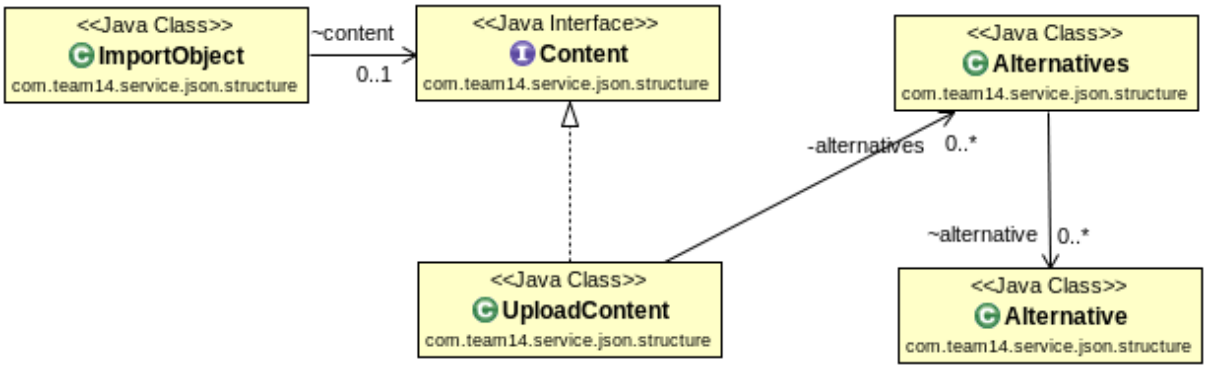
DataContext – Trieda, ktorá obsahuje inštanciu knižnice na extrakciu dát z byte-ového poľa, pričom samotná funkcionalita sa mení podľa toho, aký typ inštancie bol poslaný do konštruktora.

DataProcessor – Interface, ktorý zaručuje, že každá trieda na spracovanie špecifického typu súboru bude obsahovať metódu *getData.*

CsvProcessor - Ak sa v triede "DataConversion" zistí, že uploadnutý súbor je CSV, tak sa pomocou tejto triedy uloží do vnútornej reprezentácie pomocou frameworku Jackson.

XlsProcessor - Ak sa v triede “DataConversion” zistí, že uploadnutý súbor je XLS, tak sa pomocou tejto triedy uloží do vnútornej reprezentácie pomocou frameworku Apache POI a tiež sa zvlášť uloží prvý riadok súboru ako pri CSV.

XlsXProcessor - Podobný princíp ako pri XlsProcessor.



Obrázok 6: Formát odoslaného JSON-u

Content – Interface pre rôzne typy formátu JSON-u, ktorý backend vracia ako odpoveď na dopyty

UploadContent – Trieda špecifikujúca obsah objektu „Content“ v štruktúre odpovede z endpointu na nahratie súboru

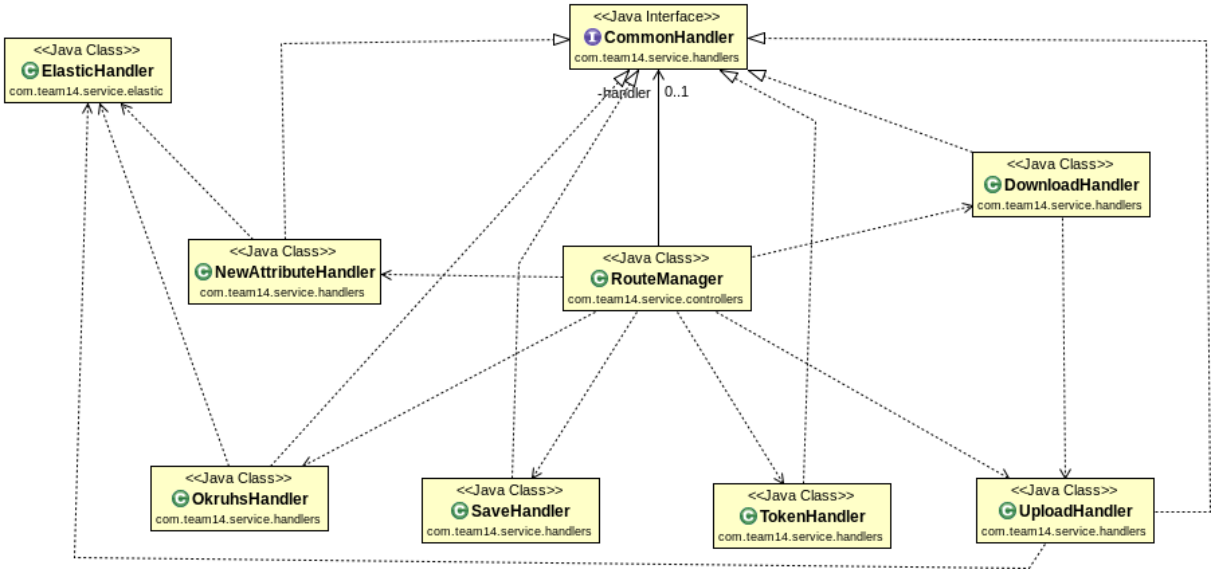
Alternative – Trieda špecifikujúca informácie o alternatívnom názve stĺpca, spolu s normalizovaným skóre z elastic search databázy a flagom, ktorý indikuje, či je daný názov schválený a môže sa bezpečne použiť.

Alternatives – Trieda obsahuje pole inštancií triedy Alternative.

ImportObject - Objekt, ktorý sa odosiela po uploadnutí ako JSON object, tento object obsahuje aj nasledovné objekty: Content, Alternatives a Alternative a majú nasledovnú štruktúru:

{
    "**error**" (*string*),
    "**content**" {
        "**titles**" (*array, string*),
        "**data**" (*array, array*),
        "**alternatives**" (*array*) {
            "**alternative**" (*array*) {
                "**type**" (*string*),
                "**rank**" (*float*)
            }

```
        }
    }
}
```



Obrázok 7: Handlers

ElasticHandler – Zabezpečujú komunikáciu medzi BE a databázou.

CommonHandler – Interface pre rôzne handlery, ktorý špecifikuje, že každý handler musí obsahovať metódu *handle.*

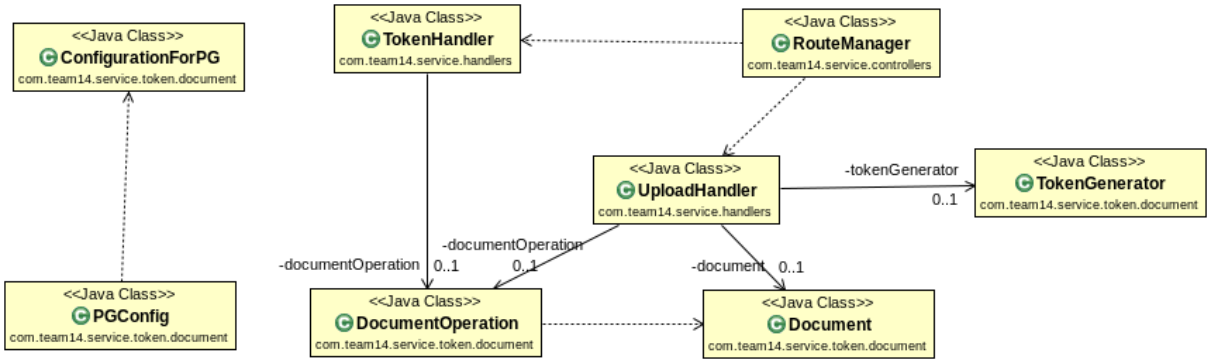
NewAttributeHandler – Táto trieda obsahuje logiku pridávania nového atribútu do elastic search databázy po zavedení nového atribútu používateľom.

DownloadHandler – Táto trieda obsahuje logiku stiahnutia súboru po ukončení zmien používateľom. Pomocou tokenu sa z PostgreSQL databázy vytiahnu informácie o súbore, ktorý je následne odoslaný na frontend.

OkruhsHandler – TOMAS BOZIK TOTO SPRAVIL

SaveHandler – Táto trieda slúži na uloženie nových názvov stĺpcov do súboru, ktoré sa z frontendu odošlú po potvrdení vykonaných zmien používateľom.

TokenHandler – Trieda obsahuje štruktúru tokenu, ktorý slúži ako identifikátor dokumentov, s ktorými používateľ pracuje.



Obrázok 8: Tokeny k dokumentom

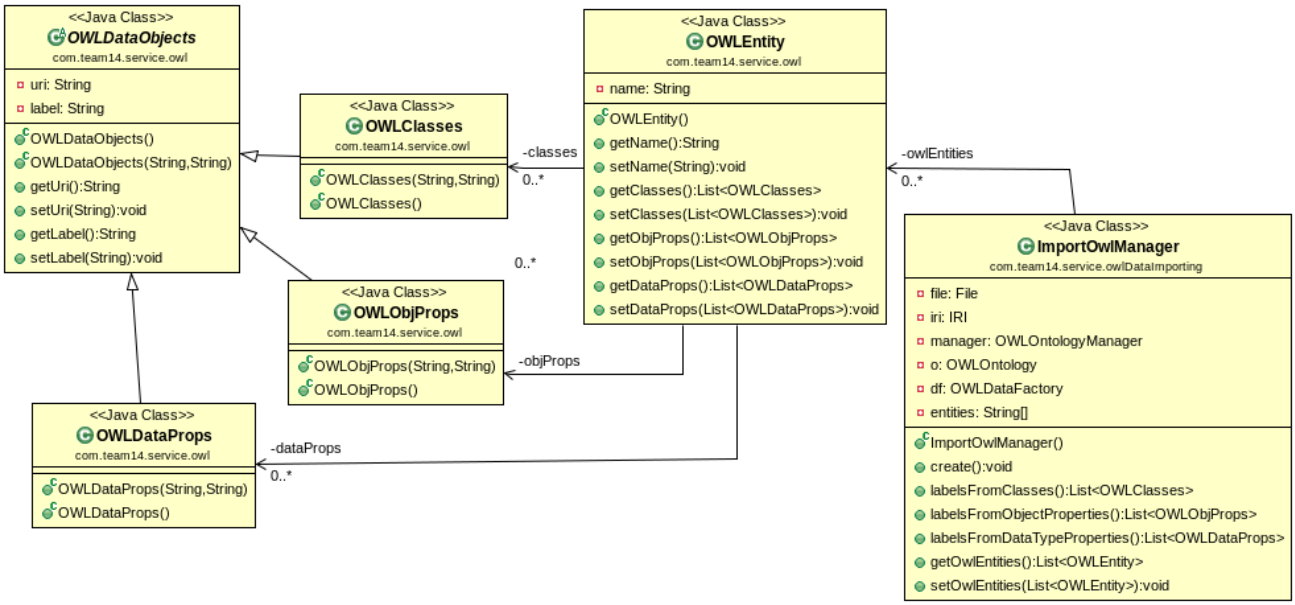
TokenGenerator – Trieda, ktorá slúži na generovanie náhodných tokenov typu 4. Vygenerovaný token je zatiaľ používaný na identifikáciu súborov, s ktorými používatelia pracujú, avšak tokeny generované touto triedou možno využiť v budúcnosti aj na iné účely.

Document – Trieda reprezentujúca štruktúru uchovávaných informácií o súboroch, ako napríklad token, názov a cesta k súboru, timestamp nahrania súboru a či je súbor upravovaný.

DocumentOperation – Táto trieda obsahuje logiku pracovania s PostgreSQL databázou na vkladanie a vyťahovanie dát o súboroch.

PGConfig, ConfigurationForPG – Triedy, ktoré slúžia na konfikuráciu a inicializáciu objektu, ktorý sa používa na komunikáciu s PostgreSQL databázou

Errors - Zachovávajú sa tu všetky chybové hlášky, ktoré sa posielajú na FE



Obrázok 9: Spracovanie centrálneho modelu

ImportOwlManager - Táto trieda spracuje celý centrálny model a spracuje ho do listu triedy OWLEntity. Následne je možné nahraťCentrálny model do databázy.

OWLEntity - OWLEntity vyjadruje jednu entitu v Centrálnom modeli, každá takáto entita obsahuje: Classes, Object properties a Data properties. Pretože má každá z týchto "tried" iné správanie, tak sme si vytvorili tri rozdielne triedy, ktoré budeme v budúcnosti využívať.

OWLClasses –

OWLObjProps –

OWLDataProps –

OWLDataObjects - Táto trieda predstavuje OWL objekt. Obsahuje URI, label (meno), kategóriu, skupinu, opis, podobné frázy .

6. Testovanie

6.1 Testovanie na Backende

V rámci testovania uplatňujeme testovanie pomocou Unit testov. Na backende sme použili na testovanie framework JUnit. Prebehlo viacero testovacích scenárov, ktoré sme si rozdelili do nasledovných kategórií:

1. Testy na správne spracovanie:

- CSV: V tomto teste sa kontroluje či sa správne načítal csv súbor, ktorý sa z byte-ového poľa uloží do vnútornej reprezentácie pomocou frameworku Jackson.
- XLS: V tomto teste sa kontroluje či sa správne načítal xls súbor, ktorý sa z byte-obého poľa uloži do vnútornej reprezentácie pomocou frameworku Apache POI.
- XLSX: Podobný test ako pri XLS, len na súbor typu xlsx.

Tiež sa tu kontroluje či sa správne zvlášť uložila hlavička zo súborov (prvý riadok) a zvlášť dáta za súborov.

2. Testy na správne určenie typu nahratého súboru: V tomto teste sa skontroluje či sa správne určili typy súborov pre csv, xls a xlsx.

3. Test na pripojenie do Elasticsearch a vyhľadanie alternatív pre testovací súbor: V tomto teste sa testuje pripojenie k Elasticsearch a následné vyhľadanie názvov stĺpcov z testovacieho súboru v Elasticsearch.

4. Test na OWL: V tomto teste sa testuje či sa načítali všetky typy entít z OWL súborov.

5. Test na pripojenie do PostgreSQL – V teste sa testuje pripojenie na PostgreSQL a pridanie nového záznamu.

Ako separátny test sme vytvorili testovací skript v jazyku Python. V repozitári tohto skriptu je zložka, ktorá obsahuje skupiny testovacích súborov, organizovaných tak, že každá skupina je v samostatnej zložke a musí obsahovať súbor, ktorého názov končí slovom *original*. Tento súbor je vo formáte, ktorý je ideálny pre publikáciu. Ostatné súbory v zložke vychádzajú z tohto súboru, ale obsahujú určité chyby, ktoré má byť naša aplikácia schopná riešiť.

Po spustení skript prejde všetky zložky, načíta dáta zo súboru so správnou formou dát a tie porovnáva s dátami, ktoré vráti naša aplikácia po spracovaní dát z ostatných súborov. Následne vypíše sumár výsledkov. Pre hlbšie preskúmanie výsledkov je vhodná verzia v

jupyter notebook-u, kde sa dá zavolať funkcia na vypísanie všetkých dát, ktoré naša aplikácia nedokázala správne spracovať.

6.2 Testovanie na Frontende

Kedže na frontende používame framework Angular 4, ktorý je populárny, veľmi používaný a existuje k nemu množstvo nástrojov na testovanie, tak sme sa rozhodli použiť Jasmine, testovací nástroj. Tento nástroj sme si zvolili hlavne kvôli jeho výbornej kompatibilite s angularom. Výhodou je takisto samostatné testovanie jednotlivých komponentov a oddelenie ich závislostí. To znamená, že každý komponent je vytvorený samostatne a následne sú otestované všetky funkcie, ktoré obsahuje. Jasmine podobne ako Angular vyrieši všetky nevyhnutné závislosti testovaného komponentu, čo je nesmierne užitočné. Každá testovaná funkcia je vždy zavolaná a v prípade funkcií s argumentmi sú s nimi aj zavolané a následne je buď otestovaný výstup funkcie alebo zmena stavu komponentu, v ktorom sa funkcia nachádza.

Referencie:

[1] G. Zhou and J. Su, “Named entity recognition using an HMM-based chunk tagger,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*, 2001, p. 473.

[2] J. R. Finkel and C. D. Manning, “Joint Parsing and Named Entity Recognition.”

[3] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang, “Named Entity Recognition through Classifier Combination.”