Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16, Bratislava 4

Deep Search
Dokumentácia k tímovému projektu
(Inžinierske dielo)

Vedúci tímu: Ing. Nadežda Andrejčíková, PhD.
Členovia tímu: Bc. Peter Berta, Bc. Matej Adamov, Bc. Michal Krempaský, Bc. Bronislava Pečíková, Bc. Ondrej Hamara
Akademický rok : 2017/2018

Obsah

1 Úvod

Informačná explózia so sebou prináša aj viacero problémov. Napriek tomu, že v dnešnej dobe si nemožno sťažovať na nedostatok informácií, máme často problém nájsť to, čo práve potrebujeme. Väčšina dokumentov je totiž v neštruktúrovanej podobe a získať z nich informácie typu: kto v danom období pôsobil v určitom regióne je prakticky nemožné. Fulltextové vyhľadávanie má jeden vážny nedostatok a to, že nezohľadňuje sémantiku daných kľúčových slov. V našom projekte sa zameriavame na spracovanie prirodzeného jazyka a stanovili sme si pomerne ambiciózny cieľ, ktorým je extrakcia štruktúrovaných dát z neštruktúrovaného textu so zachytením ich významu. Zameriavať sa budeme najmä na životopisy, z ktorých budeme môcť extrahovať informácie o tom kde a kedy dané osoby študovali, prípadne pôsobili. Pokúsime sa tiež z textov získať vzťahy typu kolega alebo spolužiak. Našou úlohou bude teda rozpoznávať, a pokiaľ to bude možné aj jednoznačne identifikovať, entity typu osoba, korporácia, geografická jednotka, datácia a zároveň aj identifikovať udalosť štúdium, prípadne pôsobenie a v rámci nich vzťahy medzi týmito entitami. Cieľom je tieto údaje uložiť v štruktúrovanej podobe tak, aby bolo možné v nich vyhľadávať a získať informáciu o tom kto a kedy na danom mieste študoval, s kým sa mohol poznať a vytvárať tak aj virtuálne komunity pre určité zameranie.

2 Globálne ciele projektu

Cieľom je extrakcia dát z neštruktúrovaných životopisov a ich následné uloženie do grafovej databázy. Je potrebné identifikovať nasledovné vzťahy:

- Štúdium - dátum, inštitúcia, miesto, profesori, spolužiaci
- Zamestnanie - dátum, inštitúcia, miesto, kolegovia
- Členstvo - dátum, inštitúcia, miesto, kolegovia

V ideálnom prípade by bolo vhodné taktiež vizualizovať súvislosti medzi osobnosťami pomocou grafu. Jednotlivé vrcholy budú predstavovať osobnosti a hrany budú predstavovať súvislosti medzi nimi.

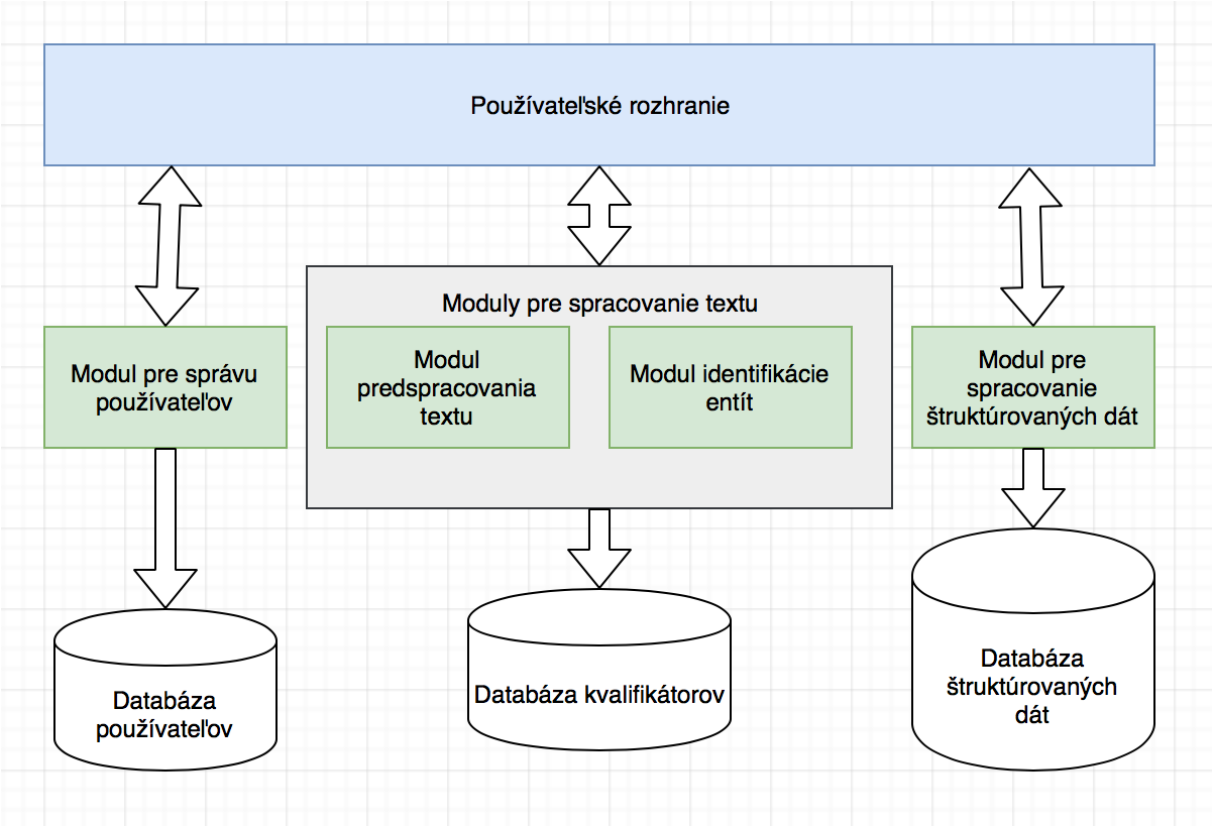
3 Celkový pohľad na systém

Systém pozostáva z piatich modulov:

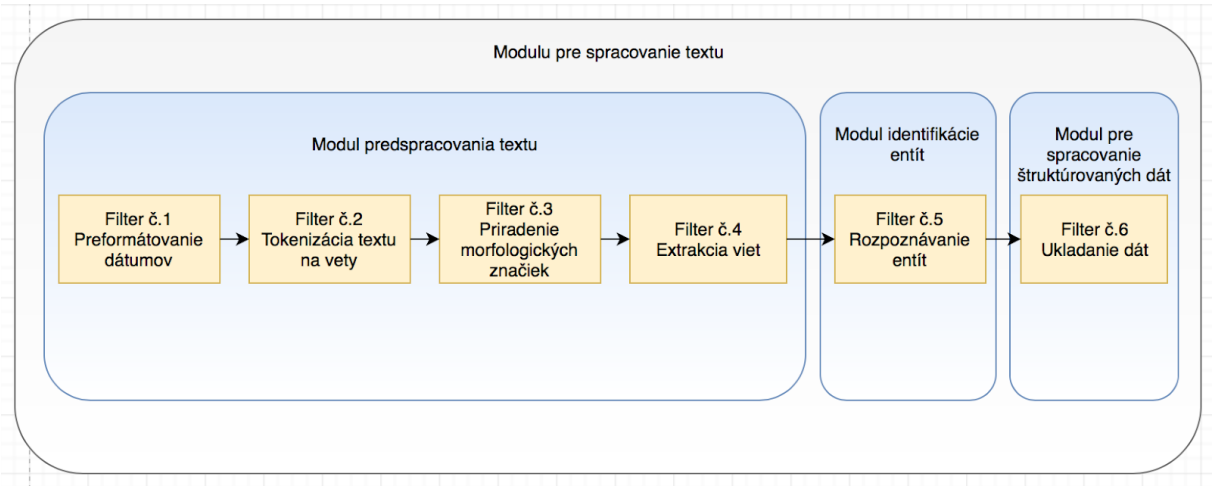
- modul používateľského rozhrania
- modul správa používateľov
- modul predspracovania textu
- modul identifikácia entít
- modul pre spracovanie štruktúrovaných dát

Modul pre používateľské rozhranie je implementovaný prostredníctvom webovej aplikácie vo frameworku Django a umožňuje používateľovi vyhľadávanie v štruktúrovaných dátach, zobrazenie grafu vzťahov, pridanie životopisu a mnohé ďalšie funkcionality (Obr. 3). Podrobnosti implementácie používateľského rozhrania sú rozpracované v kapitole Modul používateľské rozhranie. Modul správa používateľov bude ponúkať metódy ako sú prihlásenie, odhlásenie a registráciu používateľa. Najdôležitejšími modulmi systému sú nepochybne modul predspracovania textov a modul identifikácie entít. V implementácii spomínaných dvoch modulov bol využitý architektonický štýl dátovody a filtre (Obr.2).

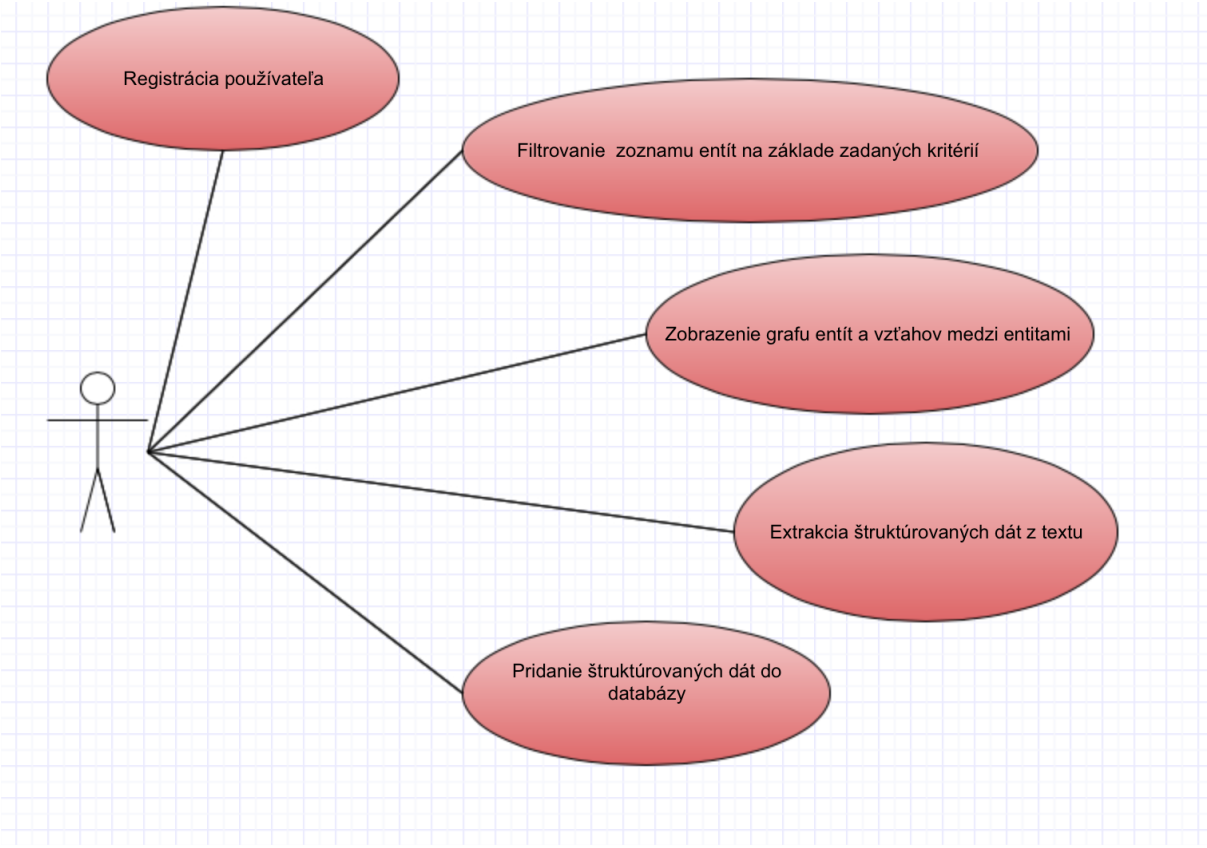
Modul pre spracovanie štruktúrovaných dát pokrýva funkcionalitu spojenú s ukladaním štruktúrovaných dát a dopytovaním sa nad týmito dátami a to najmä pre účely ich vizualizácie na používateľskom rozhraní (teda webovej aplikácii). Okrem toho bude tento modul zabezpečovať aj kompatibilitu s CIDOC CRM. Vzájomná interakcia uvedených modulov je zobrazená v diagrame (Obr.1).



Obrázok 1: Moduly systému



Obrázok 2: Architektonický štýl dátovody a filtre v projekte Deepsearch



Obrázok 3: Diagram prípadov použitia

3.1 Analýza vstupov

V rámci analýzy vstupných textov sme identifikovali kvalifikátory pre štúdium a zamestnanie. Manuálne sme spracovali približne 100 životopisov, v ktorých sme zachytili rôzne vetné skladby v okolí identifikovaných kvalifikátorov. Z tejto množiny sme vytvorili univerzálnu štruktúru pre zachytenie výskytu kvalifikátorov spolu s pomenovanými entitami v ich okolí.

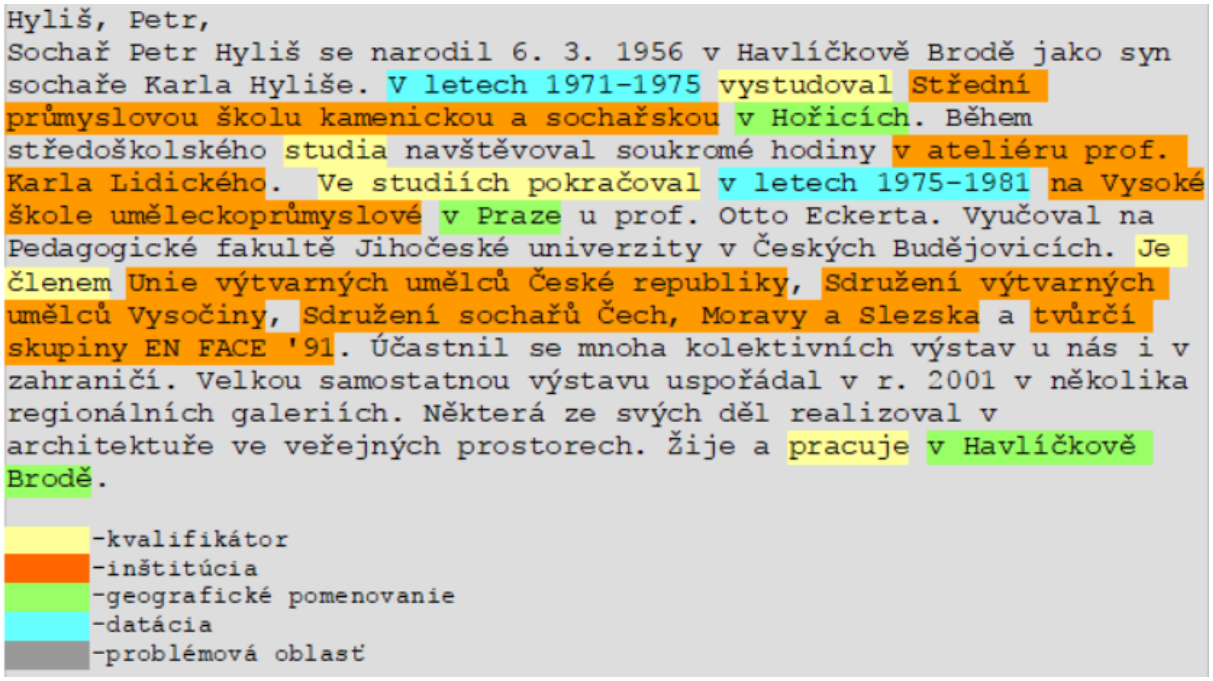
Pri zaznamenávaní pomenovaných entít sme použili dvojstupňovú kvalifikáciu pomenovaných entít pre český jazyk z článku “Czech Named Entity Corpus and SVM-based Recognizer”.

Skratky znázorňujú entity ktoré sa môžu nachádzať v okolí kvalifikátora, pričom čísla znázorňujú interval vzdialeností tejto entity od pozície kvalifikátora.

Štúdium	Zamestnanie
studovat <IC -1 +2> <GU +2> <TY -1 +2> vystudovat <IC -1 +2> <GU +2> <TY -1 +2> absolvovat <IC +1> <GU +2> <TY -1> absolvováni <IC +1> <GU> <TY> být žák <IC -1> <GU -2 +3> <TY -3 +4> studium <IC +2> <GU +1 +3> <TY -1 +1> nastoupit <IC +1> <GU +2> <TY -1> vzdeláni <IC +1> <GU +3> <TY +2>	pracovat <IC +2> <GU +2> <TY -1> být pracovník <IC> <GU> <TY> <IF +1> působit <IC -1 +1> <GU +3> <TY -1 +1> <IF> člen <IC +1> <GU +2> <TY -1> <IF> zakladatel <IC +1> <GU -1 +1> <TY> <IF> spoluzakladatel <IC +1> <GU +2> <TY +3> <IF> zamestnanec <IC> <GU> <TY -1> <IF +1>

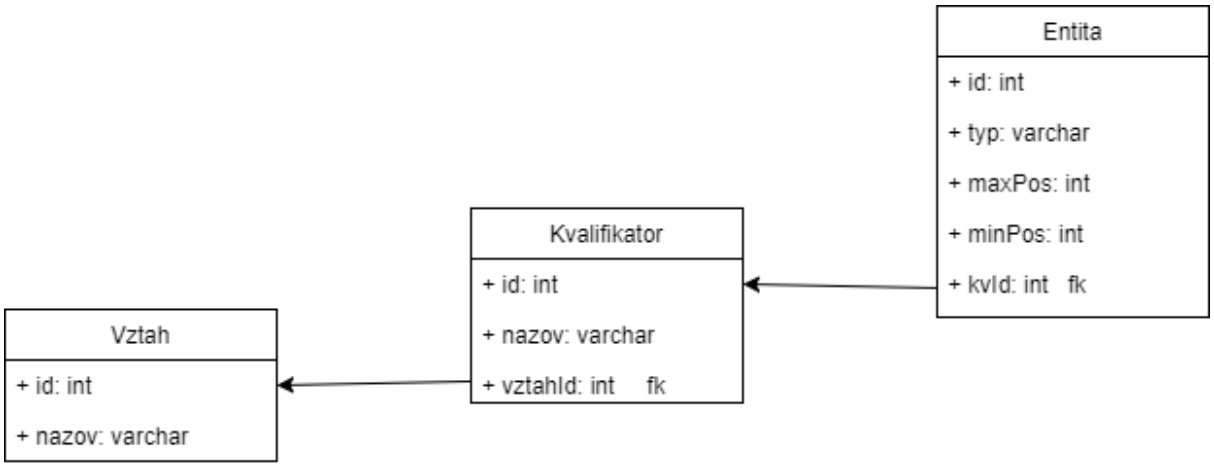
Tabuľka 1: Kvalifikátory

Túto štruktúru sme manuálne simulovali nad množinou nových životopisov a výsledky sme v nich farebne vizualizovali. Táto simulácia dosiahla 80% úspešnosť v rámci zachytenia kvalifikátora definovanou štruktúrou.



Obrázok 4: Simulácia

Vzhľadom na výsledky simulácie tejto štruktúry, sme navrhli model relačnej databázy, ktorý sme následne implementovali, a naplnili doposiaľ identifikovanými kvalifikátormi.



Obrázok 5: Relačný model

3.2 Výber databázy pre ukladanie neštruktúrovaných dát

Pre ukladanie neštruktúrovaných dát sme zvažovali tri alternatívy: ukladanie v databáze Elasticsearch, PostgreSQL alebo ukladanie životopisov v textových súboroch. Z dôvodu nedostatku pamäte na serveri sme alternatívu Elasticsearch museli vylúčiť a napokon sme sa rozhodli pre PostgreSQL nakoľko v tejto databáze máme uložené aj kvalifikátory a textové súbory by v budúcnosti nemuseli byť postačujúce.

V rámci analýzy dostupných možností pre ukladanie neštruktúrovaných dát sme vykonali aj overenie koncepcie pre identifikáciu a pomenovanie entít prostredníctvom databázy Elasticsearch. Na základe tejto analýzy sme vyhodnotili, že databáza Elasticsearch nie je pre tento účel postačujúca, nakoľko jej posledná verzia neobsahuje funkcionalitu potrebnú pre identifikáciu a pomenovanie entít.

3.3 Modul predspracovania textu

3.3.1 Požiadavky

Za účelom extrakcie poznatkov z vety je potrebné vytvoriť niekoľko nástrojov umožňujúcich zmysluplné dopytovanie nad textom. Pre tieto účely potrebujeme niekoľko metód, ktoré budú schopné spracovávať text a vytvárať množinu poznatkov o slovách vo vetách. Potrebné je vykonať nasledovné kroky:

- Tokenizácia - text potrebujeme vo forme tokenov - samostatných slov a viet
- Lematizácia - potrebujeme určiť základný tvar slov aby sme mohli vyhľadávať kvalifikátory
- Pridelenie morfologických značiek - je potrebné každému slovu prideliť morfologické značky, ktoré nám následne umožnia identifikovať jeho slovný druh.

3.3.2 Analýza prístupov cez jednotlivé služby

Pre väčšinu programovacích jazykov existuje podpora prístupov k slovníkom obsahujúcim tieto vedomosti prostredníctvom webových služieb alebo voľne dostupných knižníc. V nasledujúcej časti je uvedená analýza dostupných knižníc a webových služieb pre spracovanie českého jazyku:

3.3.2.1 NLTK

Natural Language Tool Kit - je knižnica v Pythone, určená primárne pre účely spracovávania anglického textu. Obsahuje mnoho slovníkov a funkcionalít, slúžiacich na generovanie tagov, lem, identifikáciu slovných druhov alebo tokenizáciu.

3.3.2.2 Lemmagen

Jedná sa o český voľne dostupný morfologický slovník a značkovač určený pre spracovávanie textu v českom jazyku. Väčšina služieb tejto knižnice vykonáva morfologickú analýzu nad vloženým textom a ako výstup poskytuje morfologické značky, tokeny a lematizované formy slov na základe natrénovaného lingvistického modelu.

3.3.2.3 MorhpoDiTa

Táto webová služba je dostupná aj cez rozhranie REST API, ktoré nám umožňuje na základe metód tokenize, analyze a tag vrátiť nad poslaným textom výstup vo forme JSON, kde je každé slovo tokenizované s pridelenou lemou a morfologickou značkou.

Ponúka webové rozhrania s REST API ako aj knižnicu v C++ na lokálne použitie. Príklad použitia na webe je zobrazený na obrázku 6.

Obrázok 6: Príklad použitia na webe

```
model:              "czech-morfflex-pdt-161115"
acknowledgements:
  0:                "http://ufal.mff.cuni.cz/morphodita#morphodita_acknowledgements"
  1:                "http://ufal.mff.cuni.cz/morphodita/users-manual#czech-morfflex-pdt_acknowledgements"
result:
  0:
    0:
      token:        "Děti"
      analyses:
        0:
          lemma:    "dítě"
          tag:      "POS=N|SubPOS=N|Gen=F|Num=P|Cas=1|Neg=A"
        1:
          lemma:    "dítě"
          tag:      "POS=N|SubPOS=N|Gen=F|Num=P|Cas=4|Neg=A"
        2:
          lemma:    "dítě"
          tag:      "POS=N|SubPOS=N|Gen=F|Num=P|Cas=5|Neg=A"
      space:        " "
    1:
      token:        "pojedou"
      analyses:
        0:
          lemma:    "jet"
          tag:      "POS=V|SubPOS=B|Num=P|Per=3|Ten=F|Neg=A|Voi=A"
      space:        " "
```


Obrázok 7: Príklad použitia metódy analyze


```
model:              "czech-morfflex-pdt-161115"
acknowledgements:
  0:                "http://ufal.mff.cuni.cz/morphodita#morphodita_acknowledgements"
  1:                "http://ufal.mff.cuni.cz/morphodita/users-manual#czech-morfflex-pdt_acknowledgements"
result:
  0:
    0:
      token:        "Děti"
      lemma:        "dítě"
      tag:          "NNFP1-----A----"
      space:        " "
    1:
      token:        "pojedou"
      lemma:        "jet-1_^(pohybovat_se,_ne_však_chůzí)"
      tag:          "VB-P---3F-AA---"
      space:        " "
    2:
      token:        "k"
      lemma:        "k-1"
      tag:          "RR--3----------"
      space:        " "
    3:
      token:        "babičce"
      lemma:        "babička"
      tag:          "NNFS3-----A----"
    4:
      token:        "."
      lemma:        "."
      tag:          "Z:-------------"
      space:        " "
```


Obrázok 8: Príklad použitia metódy tag

3.3.2.4 KonText

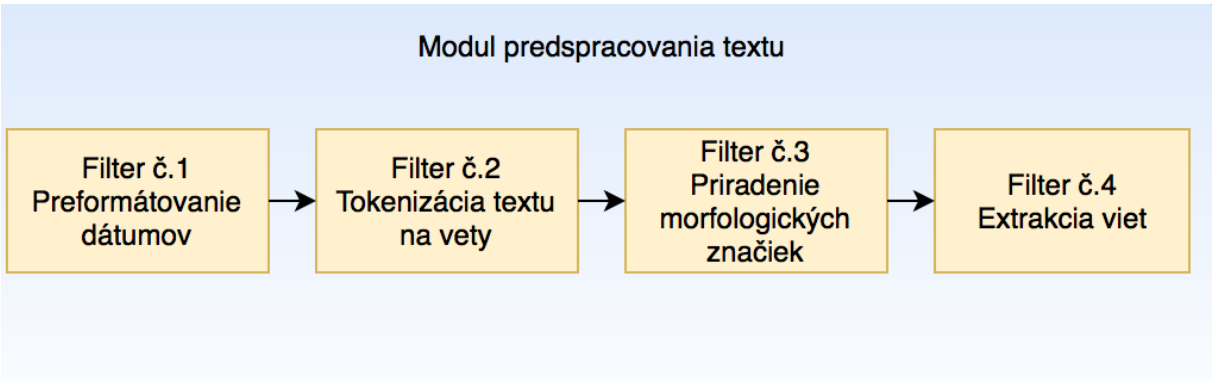
Jedná sa o rozhranie umožňujúce pristupovať ku slovníkom alebo korpusom českého jazyka. Každý z týchto korpusov reprezentuje vedeckú prácu akademikov českých univerzít. Tie následne poskytujú niekoľko operácií nad svojimi korpusmi. Nevýhoda uvedeného prístupu je, že nie je dostupná knižnica alebo prístup, ktorý by nám jednoducho umožnil lokálne sa dopytovať nad týmto rozhraním. Niektoré operácie dokonca vyžadujú prihlasovanie sa na webovej stránke. Tento koncept považujeme za nevýhodný a pokúsime sa nájsť alternatívne prístupy.

3.3.2.5 Text.fiit

Jedná sa o stránku výskumu fakulty FIIT, ktorá sa zaoberá spracovaním slovenského jazyka. Na tejto stránke máme k dispozícií metódy tokenizácie, lematizácie, morfologických značiek a určovania slovných druhov. Tieto metódy sú prístupné vo forme webových služieb a tento prístup sme zhodnotili ako vhodný pre spracovávanie slovenského jazyka.

3.3.3 Návrh

Z analýzy vyplýva možnosť použitia viacerých prístupov. Ako najvhodnejšie riešenie pre účely našej práce sme identifikovali MorphoDiTu. Nakoľko očakávame rôznu postupnosť procesov spracovávania textu, je vhodné navrhnúť volanie týchto služieb v samostatných filtroch. Tento prístup nám umožní vyššiu flexibilitu definovania procesu spracovávania textu. Pre tento účel sa javí vhodné použitie architektonického štýlu dátovody a filtre, ktorý umožňuje rýchlu adaptáciu a konfiguráciu procesu spracovávania textu. Architektúru modulu si môžeme pozrieť na nasledujúcom obrázku.



Obrázok 9: Architektúra modulu predspracovania textu

Modul pozostáva zo štyroch filtrov:

- Preformátovanie textu
- Tokenizácia textu na vety
- Priradenie morfologických značiek
- Extrakcia viet

Filter č.1 má za úlohu identifikovať v životopisoch datácie vo formáte yyyy-yy (napríklad 1990-93) a prepísať ich na formát yyyy-yyyy (napríklad 1990-1993). Filter č.2 rozdeľuje text na vety, filter č.3 tokenizuje a lematizuje text a prideľuje tokenom morfologické značky. Filter č.4 extrahuje vety, ktoré obsahujú kvalifikátory.

3.3.4 Implementácia

Realizácia návrhu prebehla v duchu architektonického štýlu dátovody a filtre. Vstupné dáta spracujeme vo forme "slovníka" a následne aplikujeme metódu, ktorá sa správa ako filter a spracuje tento text. Výstup použijeme ako vstup do nasledujúceho filtra.

3.4 Modul identifikácie entít

Tento modul implementuje identifikáciu pomenovaných entít v predspracovanom texte. Identifikácia entít je realizovaná prostredníctvom nástroja NameTag a následne doplňujúcou identifikáciou a pomenovaním entít, ktoré nástroj nebol schopný rozpoznať.

- Vstupy - Vstupom pre tento modul je JSON, ktorý uchováva záznamy viet pre každý typ vzťahu. Každý záznam jednotlivej vety obsahuje pôvodné znenie vety a pozíciu kvalifikátora, ktorý sa v nej vyskytuje.

```
[{
        "relation": "študent",
        "sentences": [{
                "sentence": "Absolvoval Gymnázium M. M. Hodžu v Liptovskom Mikuláši (1985-1989)
                a Univerzitu Mateja Bela v Banskej Bystrici (slovenský jazyk – dejepis) (1989-1995).",
                "position": 0
        }]
},{
        "relation": "zamestnanec",
        "sentences": []
}]
```

- Výstupy – Dáta sú po samotnom procese identifikácie entít vo forme tripletov. Tie obsahujú subjekt (v našom prípade meno osoby), predikát (vzťah), a následne objekt v podobe identifikovanej pomenovanú entity, ktorá sa skladá z názvu, pozícií a typu. Následne sa spracujú dátumy, ktoré sa priradia k príslušným vzťahom na základe vzdialenosti od susedných entít v danej vete. Tieto datácie sa následne priradia ako parameter predikátu v konkrétnom triplete.

3.5 Modul pre spracovanie štruktúrovaných dát

3.5.1 Výber databázy pre ukladanie štruktúrovaných dát

Štruktúrované dáta získané zo spracovania životopisov je potrebné v nejakej forme ukladať. K dispozícii sú grafové databázy, objektové alebo klasické relačné databázy. Výhodou grafových databáz je, že vedia dobre reprezentovať vzťahy medzi objektami. V našom prípade by to bolo užitočné pre ukladanie vzťahov ako sú kolega, zamestnanec, spolužiak. Z grafových databáz sme zvažovali najmä Neo4j a Cayley. Nakoľko naše výstupy sú vo forme tripletov, zvažovali sme aj zakomponovanie diplomovej práce Martina Habdáka, ktorá s dátami v tomto formáte pracuje.

3.5.1.1 Neo4j

Jedna z najpopulárnejších open-source databáz, Neo4J je vyvíjaná v jazyku JAVA. Táto databáza umožňuje vizualizáciu dát na základe dopytov v jazyku cypher. Neo4j má široké využitie, napríklad v telekomunikačných službách pri detekcii fraudov, sieťových a IT operáciách atď.

Výhody:

- dlho žijúci projekt s veľkou komunitou
- podpora pre veľké množstvo jazykov
- end to end control - napr. nie je závislá na externých úložiskách ako Cayley
- RESTful api

Nevýhody:

- nie je cloud based (nemusí sa vždy jednať o nevýhodu)
- nepodporuje shardovanie
- spoplatnená pre komerčné použitie

3.5.1.2 Cayley

Cayley je neoficiálny opensource produkt od spoločnosti Google vyvíjaný v jazyku Go, inšpirovaný grafovou databázou Freebase. Cieľom databázy Cayley je vytvorenie tzv. toolboxu pre pracovanie s prepojenými resp. grafovými dátami (sémantický web, sociálne sieťe, atď.).

Výhody:

- podporuje viac query jazykov ako napr. Gizmo, GraphQL alebo MLQ
- zabudovaný query editor a vizualizér
- RESTful API
- podpora viacerých backend úložísk: kvl (Bolt, LevelDB), NoSQL (MongoDB), SQL (Postgres, CocroachDB, MySQL)
- modulárny design (jednoduché rozširovanie s novými jazykmi a backend riešeniami)
- veľký dôraz na rýchlosť

Nevýhody:

- pomerne mladý projekt - ku dnešnému dňu má iba 1130 commitov a 59 prispievateľov
- neúplná dokumentácia
- storage rieši v externých databázach (môže byť aj výhoda)e

3.5.1.3 Záver

Rozhodli sme sa pre databázu Neo4j nakoľko bolo pre nás kľúčové aby databáza bola dobre zdokumentovaná. Žiaden z členov tímu nemá skúsenosti s prácou so žiadnou z grafových databáz. Pre učenie sa novej technológie je veľkou výhodou jej dobrá dokumentácia. Ak by sme zvolili napríklad Cayley mohlo by to ohroziť dodržanie termínov odovzdania, nakoľko by sme sa nemuseli stihnúť túto technológiu dostatočne rýchlo naučiť.

3.6 Výber Neo4j Python driveru

Počas výberu technológie na prístup do Neo4j databázy z jazyku Python sme analyzovali niekoľko knižníc. Priamo na webovej stránke Neo4j je uvedených niekoľko odporúčaných knižníc.

Neo4j v dokumentácií uvádza základnú podporu pre knižnicu Neo4j Python Driver, ktorá natívne podporuje okrem http a https portov aj bolt port.

Z tejto základnej knižnice ďalej existuje viacero komunitných riešení. Medzi ne patria Py2neo, Neomodel, Neo4jRestClient. V nasledujúcej tabuľke je uvedené porovnanie spomínaných knižníc.

Knižnica	Python 3.6 podpora	Neo4j 3.0 podpora	Aktivita na git-e	Výhody	Nevýhody
Py2neo	3.5, ale asi funguje aj na 3.6	Áno	1 rok Core údržba	OGM Dobrá dokumentácia	OGM dokumentácia ma málo príkladov
Neomodel	3.3+	Áno	4 mesiace Core údržba	Existuje Django verzia, jednoduché použitie OGM Aktívny vývoj	Málo commitov
Neo4j RestClient	Skôr nie, ale Python 3 áno	Áno	4 roky Core údržba	OGM, celkom pekné	Nie aktívny vývoj

Tabuľka 2: Porovnanie knižníc na pripojenie do Neo4j

Pre účely integrácie systému s databázou Neo4j sme zvolili driver Neomodel. Nakoľko tento ovládač poskytuje všetku funkcionalitu potrebnú pre účely vytváraného produktu a prívetivé OGM.

3.7 Dátový model

Okrem vhodnej prezentácie extrahovaných dát sme si dali za cieľ aj to, aby bol vytváraný systém kompatibilný s CIDOC CRM. Z tohoto dôvodu sme sa rozhodli vytvoriť v úložisku pre štruktúrované dáta dva modely - jeden pre účely prezentácie štruktúrovaných dát a druhý pre účely kompatibility s ontologickým rámcom CIDOC CRM.

3.7.1.1 Prezentačný model

Úlohou prezentačného modelu je ukladať dáta vo forme čo najvhodnejšej pre ich následnú vizualizáciu vo webovej aplikácii. Tento model by mal byť vytvorený tak, aby bol čo najintuitívnejší. Cieľom je, aby sa aj laik vedel pomerne rýchlo zorientovať v prezentačnom grafe. Práve z tohoto dôvodu sme sa rozhodli, na rozdiel od CIDOC CRM modelu v prezentačnom modeli reprezentovať časovú pečiatku ako atribút vzťahu štúdia alebo zamestnania. Ďalej bude tento model obsahovať lokalizáciu, reprezentovanú vzťahom, môže ísť napríklad o osobu žijúcu, či pôsobiacu v meste, o korporáciu ktorá má sídlo v meste alebo

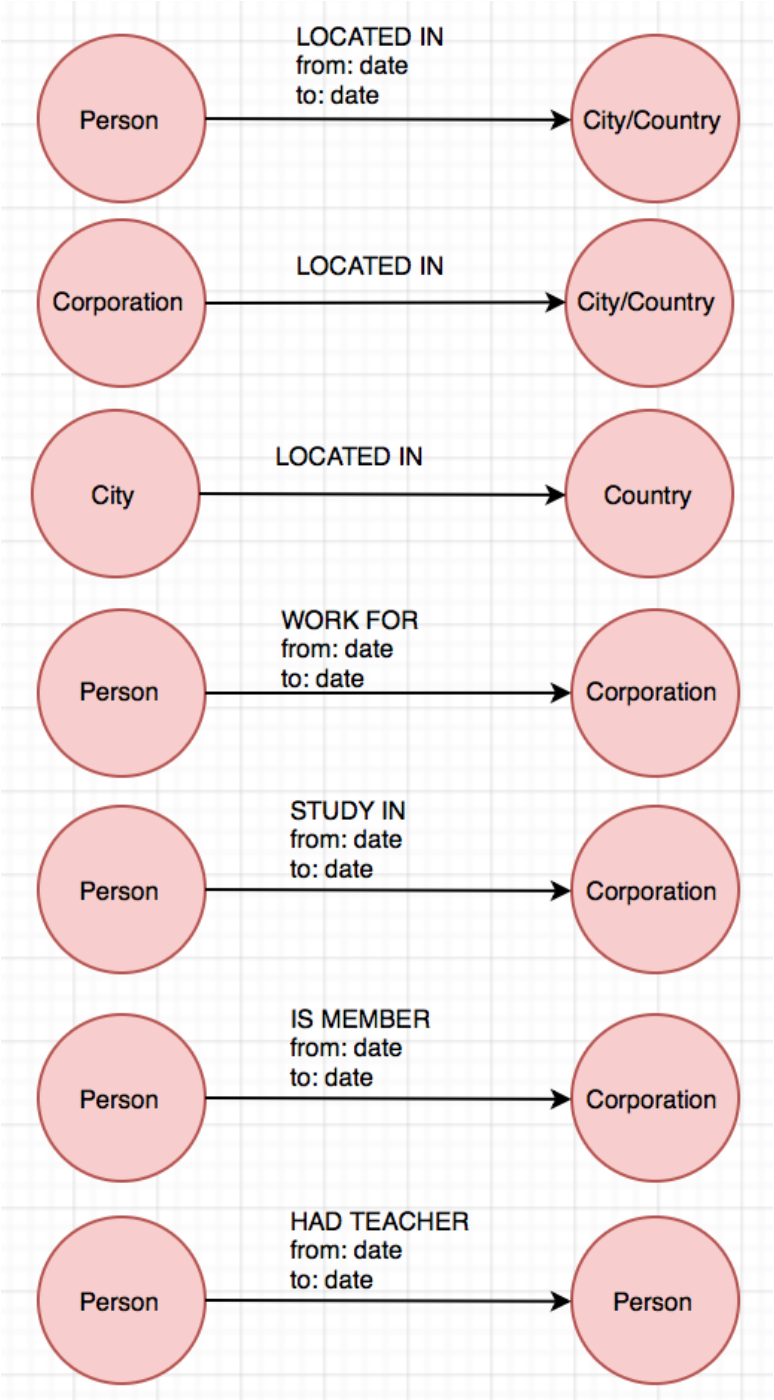
o mesto nachádzajúce sa v štáte. Prezentačný model pracuje s piatimi typmi uzlov menovite sú to:

- Person
- City
- Country
- Corporation
- Alternative name

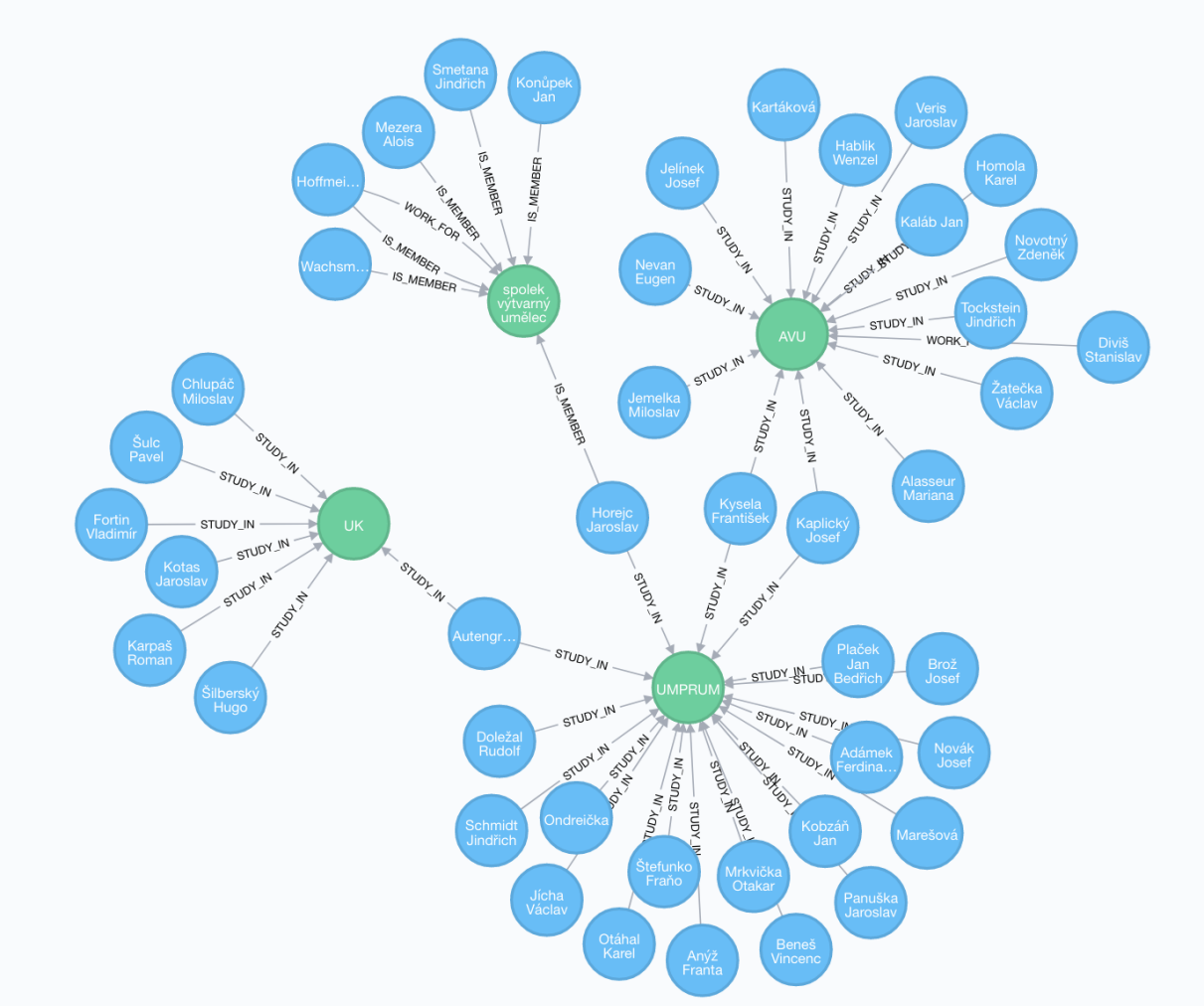
Okrem spomínaných typov uzlov tento model definuje nasledovné typy vzťahov:

- LOCATED IN
- STUDY IN
- WORK FOR
- IS MEMBER
- HAD TEACHER

Na obrázku 8 je definované, akými vzťahmi môžu byť prepojené jednotlivé typy uzlov. Vzťahy LOCATED IN, WORK FOR a STUDY IN môžu obsahovať datáciu ako atribút. Žiadne iné prepojenia v tomto modeli nie sú povolené. Príklad grafu v prezentačnom modeli môžeme vidieť na nasledujúcom obrázku 10.



Obrázok 10: Prepojenia v prezentačnom modeli



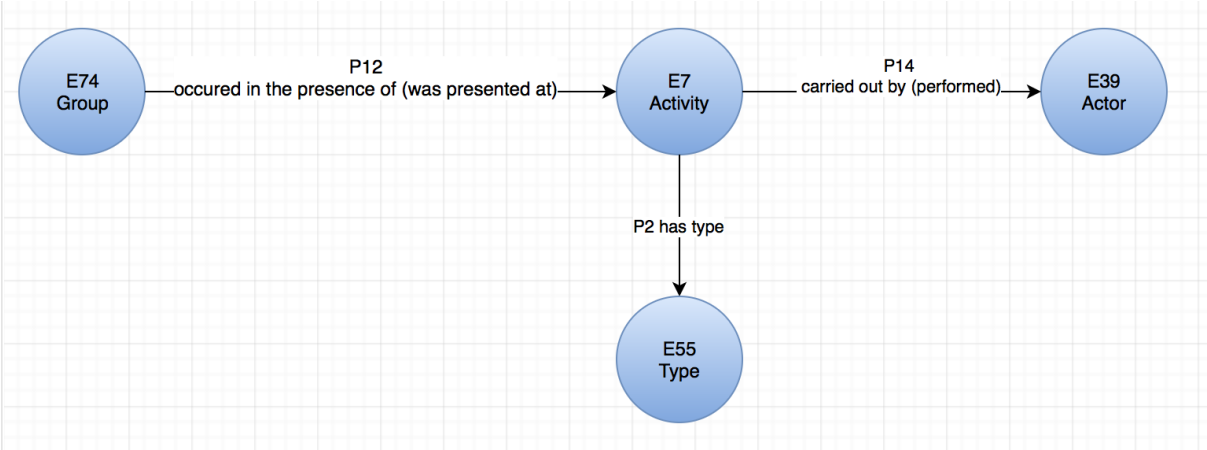
Obrázok 11: Príklad prezentačného grafu

3.7.1.2 CIDOC CRM model

Tento model bol vytvorený na základe modelu CIDOC CRM, pričom všetky CIDOC CRM entity sú reprezentované prostredníctvom rovnomenných uzlov v grafovej databáze a všetky CIDOC CRM property sú reprezentované prostredníctvom rovnomenných vzťahov v grafovej databáze.

3.7.1.3 Reprezentácia vzťahu štúdia a vzťahu zamestnania

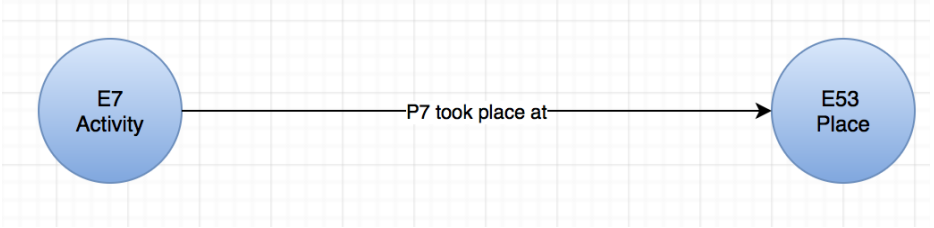
Štúdium, zamestnanie a členstvo je v modeli reprezentované entitou E7 Activity, táto je prostredníctvom vzťahu P2 has type prepojená s entitou E55 Type, ktorá špecifikuje či sa jedná o štúdium, členstvo alebo o zamestnanie. Entita E74 Group v modeli reprezentuje organizáciu a entita E39 Actor konkrétnych aktérov v danej udalosti.



Obrázok 12: Reprezentácia vzťahu štúdia a zamestnania

3.7.1.4 Reprezentácia vzťahu lokalizácie

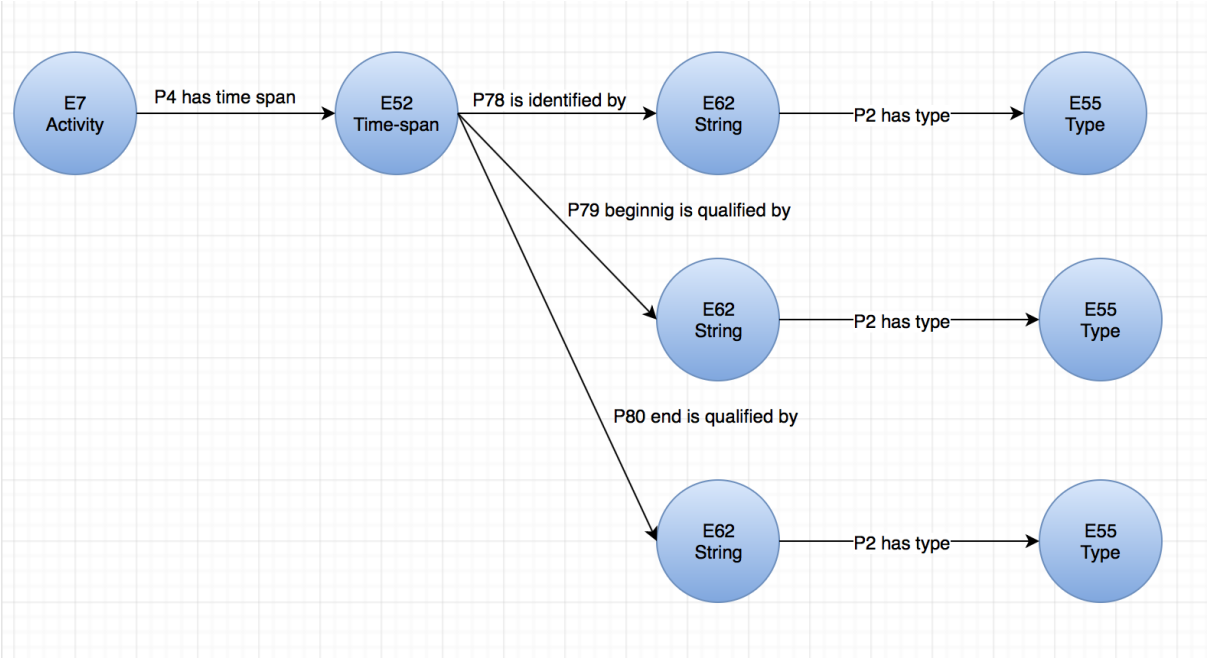
Model umožňuje každej entite E7 Activity priradiť lokalizáciu prostredníctvom vzťahu P7 took place at a entity E53 Place ako to môžeme vidieť na obrázku 13.



Obrázok 13: Reprezentácia vzťahu lokalizácie

3.7.1.5 Priradenie časovej pečiatky

Datácia je v modeli reprezentovaná entitou E52 Time-span, táto je s entitou E7 Activity prepojená prostredníctvom vzťahu P4 has time span ako to môžeme vidieť na nasledujúcom obrázku Obr. 14.



Obrázok 14: Datácia

3.8 Import dát

Pre presnejšiu identifikáciu entít a ich rozpoznanie, či určenie jednoznačnosti sa v spracovaní prirodzeného jazyka používajú rôzne metódy. Jednou z nich je slovníková metóda, ktorá pre angličtinu a širšie používané jazyky je jednoduchšie použiteľná, nakoľko je napr. pre angličtinu budovaný tzv. wordnet slovník, ktorý sa v tejto súvislosti úspešne využíva. U nás, resp. pre naše jazyky takýto slovník neexistuje, preto sme hľadali slovníky , ktoré by nám pomohli jednotlivé entity lepšie rozpoznávať. U nás overené slovníky budujú organizácie štátnej správy, ale aj knižnice a iné typy pamäťových a fondových inštitúcií. Preto sme sa rozhodli využiť databázu Národných autorít ČR , ktoré sú sprístupnené na stránkach NK ČR, resp. z projektu INTERPI. Tento projekt spracováva osoby, korporácie, geo pojmy, ale aj iné entity vrátane možných vzťahov medzi nimi navzájom. Na rozdiel od Národných autorít ČR, v rámci projektu INTERPI môžu kooperovať rôzne typy pamäťových inštitúcií bez rozdielu formátov a pravidiel, ktoré pre spracovanie takýchto dát používajú. Predstavuje platformu pre kooperatívnu tvorbu entít, používaných pre popis kultúrnych informačných objektov v pamäťových inštitúciách v heterogénnom prostredí on-line. Jednotlivým inštanciám tried, ktoré reprezentujú tieto entity, priraďuje perzistentný identifikátor a každú inštanciu je zároveň možné spracovať podľa viacerých pravidiel a v rôznych formátoch, tak ako sú tieto definované pre daný typ pamäťovej inštitúcie.

Dáta budované v rámci týchto projektov sú dostupné po registrácii a na ich stránke. Oba tieto projekty spravuje Národní knihovna ČR, ktorá zároveň poskytuje rozhranie, pomocou ktorého vieme tieto dáta získať v XML formáte. Rozhodli sme sa preto importovať si ich do nášho systému. Pri spracovaní prirodzeného ich potom využívame pre rozpoznanie pomenovaných entít.

Tieto dáta sme použili pre vstupné naplnenie grafovej databázy informáciami o geografických entitách, organizáciách a osobách. Dáta z Národných autorít ČR sú dostupné vo formáte MARC/XML, konkrétne vo formáte MARC21 pre autority a dáta z projektu INTERPI sú dostupné v XML formáte definované vlastnou XSD schémou.

3.8.1.1 Import korporácií

V MARC21 formáte vieme identifikovať korporácie prostredníctvom tagu 110. Dáta sme importovali prostredníctvom jednoduchej funkcie, ktorá rozparsovala xml a každý tag s hodnotou 110 uložila do databázy ako uzol typu korporácia, v prípade, ak sa daná korporácia ešte v databáze nenachádzal.

3.8.1.2 Import geografických entít

V MARC21 formáte vieme identifikovať geolokácie prostredníctvom tagu 370. Pričom lokácia sa v MARC21 ukladá v nasledovnom tvare: "mesto, krajina". Dáta sme importovali prostredníctvom jednoduchej funkcie, ktorá rozparsovala xml a v každom tagu s hodnotou 370 identifikovala mesto a krajinu a tieto uložila do grafovej databázy ako uzly príslušných typov.

3.8.1.3 Prepojenie entít vzťahmi

Vzťahy, ktorých identifikácia bola opísaná v kapitole 3.3. sme sa rozhodli ukladať do databázy neo4j. Ukladanie identifikovaných vzťahov do databázy je implementované vo filtri s názvom "NeoStore" a to prostredníctvom jednoduchého zavolania funkcii rozhrania databázy. Vzťahy sú ukladané do oboch zadefinovaných modelov (prezentačného aj CIDOC).

3.9 Modul správa používateľov

Naša aplikácia nezahŕňa len vyhľadávanie nad štruktúrovanými dátami, ale aj samotné pridávanie nových textov pre spracovanie naším systémom. Potrebujeme teda zabezpečiť, aby sa do systému nedostali nesprávne a nepravdivé dáta. Toto dosiahneme zavedením jednoduchej správy používateľov.

Definovali sme si 3 základné používateľské role:

- Admin

  Admin je používateľ, ktorý spravuje systém. Prideľuje práva registrovaným používateľom a validuje registráciu.

- Anonymný používateľ

  Anonymný používateľ je používateľ systému, ktorý si chce vyhľadať nejakú konkrétnu informáciu v už spracovaných štruktúrovaných dátach. Takýto používateľ nemá práva na vkladanie vlastných dát do systému..

- Registrovaný používateľ

  Registrovaný používateľ je taký používateľ, ktorý sa zaregistroval do systému, bola mu schválená registrácia adminom a následne sa prihlásil do systému. Takýto používateľ, ktorému boli nastavené práva adminom, dokáže do systému vložiť vlastné dáta, ktoré náš systém následne spracuje do štruktúrovanej podoby.

Registrácia používateľa prebieha pomocou jednoduchého formulára kde používateľ vyplní svoje meno, priezvisko, emailovú adresu a zvolí si používateľské meno a prihlasovacie heslo. Vyplnené dáta systém spracuje a vytvorí používateľovi záznam v databáze používateľov. Admin používateľovi registráciu buď schváli alebo zamietne. Pri prihlasovaní používateľ vyplní používateľské meno a heslo, systém skontroluje či existuje registrovaný používateľ s

danými prihlasovacími údajmi a na základe výsledku vyhodnotí prihlásenie ako úspešné alebo neúspešné.

3.10 Modul používateľské rozhranie

3.10.1 Analýza knižníc pre vizualizáciu dát

Predmetom analýzy bolo zdokumentovanie výhod resp. nevýhod vizualizačných knižníc, ktoré sme na základe našich technológií vybrali.

3.10.1.1 Vis.js

Ide o dynamický, browser based nástroj na vizualizovanie dát, navrhnutý tak, aby vedel spracovať veľký počet dát. Knižnica pozostáva z nasledujúcich komponentov: DataSet, TimeLine, Network, Grph2d and Graph3d. V našom projekte je možné použiť komponent Network, z dôvodu využívania technológie DB Neo4j.

Výhody:

- jednoduché a prehľadné zobrazenie dát
- rôzne spôsoby reprezentácie uzlov (fotografie, ikony, farby)
- rôzne typy hrán (šípky, prerušované hrany, krivky)
- rôzne typy názvov (zvýraznenie, farba)
- clastering, rôzne layouty, animácie

Nevýhody:

- nemožno načítať priamo Neo4j pomocou query
- potreba upraviť vstupné dáta

Ukážky je možné zobraziť na linku http://visjs.org/network_examples.html.

3.10.1.2 Popopto.js

Popoto.js je knižnica založená na JavaScripte postavená na knižnice D3.js. Popoto.js poskytuje interface na prehľadávanie grafov, v každom modernom prehliadači.

Výhody:

- generovanie interaktívneho grafového dopytovania do webovej stránky alebo webovej aplikácie, a tak vytvárať zložité Neo4j dopyty
- interaktívne prehľadanie uzlov (rozvíjanie uzlov), rôzne typy a farby uzlov
- uzly možnosť reprezentovať obrázkami (použiteľné pri CVs)
- hrany je možné pomenovať a formatovať
- automatické rozloženie uzlov po presunutí

Nevýhody:

- menej možností pri formátovaní uzlov a hrán ako má vis.js

Ukážky sú dostupné na webovej stránke http://www.popotojs.com/live/query-viewer/index.html.

3.10.1.3 Alchemy.js

Táto knižnica predstavuje aplikáciu na grafovú vizualizáciu dát na webe. Založená na knižnici 3D.js a Ladosh.js.

Výhody:

- ľahká inicializácia grafu: var alchemy = new Alchemy(Config);
- nastavenie grafu (vlastností, uzlov, hrán) pomocou premennej: Config
- načitávanie dát pomocou graphJSON

Nevýhody:

- nutnosť formatovania vstupu dát z DB neo4j do formatu graphJSON
- neprehľadné pri väčšom počte dát

Ukážky dostupné na: http://graphalchemist.github.io/Alchemy/#/examples.

3.10.1.4 Linkurious.js

Ide o JavaScriptovú knižnicu na vizualizáciu a interakciu s grafmi. Jej súčasťou je knižnica Sigma.js.

Výhody:

- Priame volanie neo4j dopytov z frontendu a tiež zo vstupu (graphJSON)
- Použiteľné v prípade nenasadenia web appky
- rôzne spôsoby a vizuálizacie
- vizualizácia až 2000 uzlov a 5000 hrán
- možnosti jednoduchého clustrovania pomocou myši (zvýšenie prehľadnosti)

Nevýhody:

- zložitejšia konfigurácia formátovania v JavaScripte

Ukážky dostupné na stránke:
https://github.com/Linkurious/linkurious.js/blob/develop/examples/index.html

3.10.1.5 KeyLines.js

Nevýhody

- nie je možná plná verzia, len skúšobná verzia na 30 dní v obmedzenej funkcionalite

3.10.1.6 Zhodnotenie

Po hĺbkovej analýze a následnom odskúšaní vyššie spomenutých knižníc sme sa rozhodli vybrať a v našej aplikácií na vizualizáciu dát použiť knižnicu vis.js. Hlavnými aspektmi výberu boli jednoduché prepojenie knižnice s neo4j databázou, ľahká konfigurácia zobrazenia grafových elementov a interaktívne prvky (popup okno, možnosť kliku, aj dvojkliku na uzol a pod.).

3.10.2 Návrh používateľského rozhrania

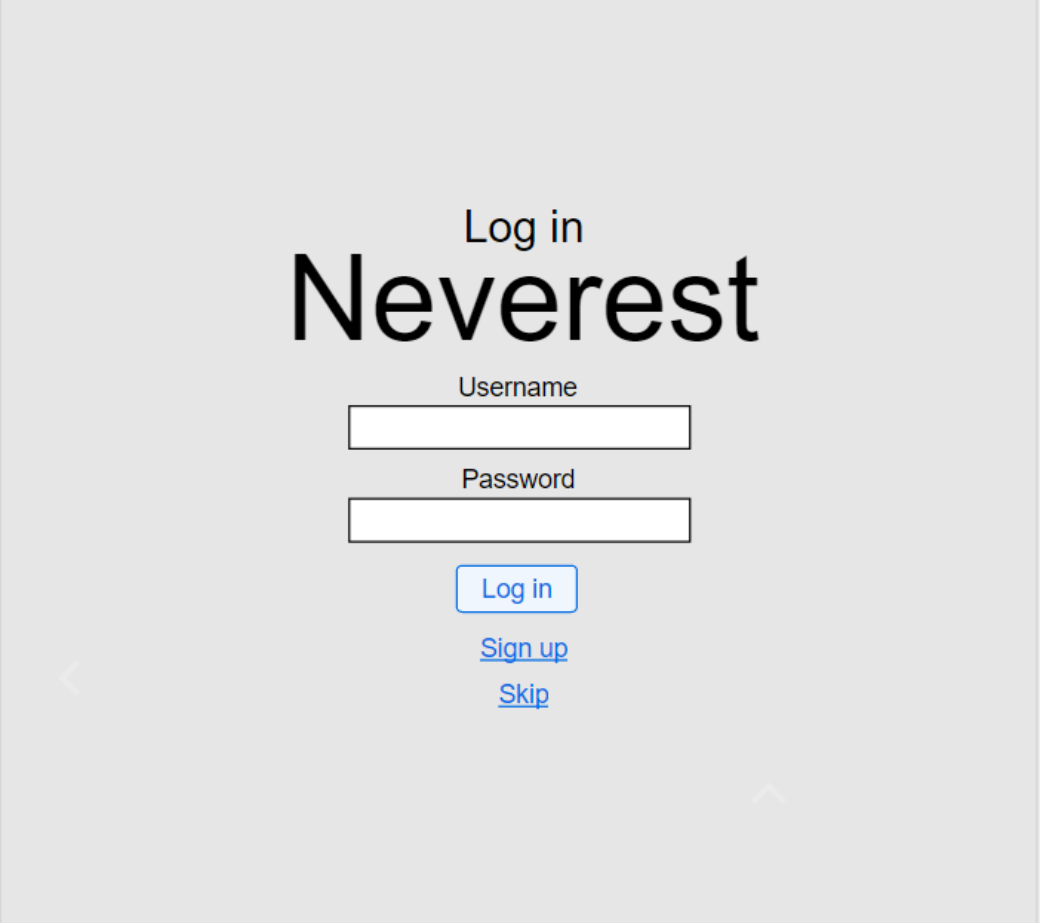
Systém bude sprístupnený používateľom v podobe webového rozhrania a bude ponúkať nasledovnú funkcionalitu:

- vyhľadávanie informácii o zadaných entitách
- filtrovanie zoznamu entít na základe zadaných kritérií
- spracovanie a zobrazenie vlastného textu do štruktúrovanej podoby
- pridanie štruktúrovaných dát do databázy

Každá funkcionalita je dostupná na základe typu účtu pod ktorým je používateľ prihlásený. Preto je potrebné vytvorenie obrazoviek na prihlasovanie a registráciu, ktoré budú spolu s modulom správy používateľov poskytovať možnosť prihlasovania a správy účtu.

3.10.2.1 Prihlásenie

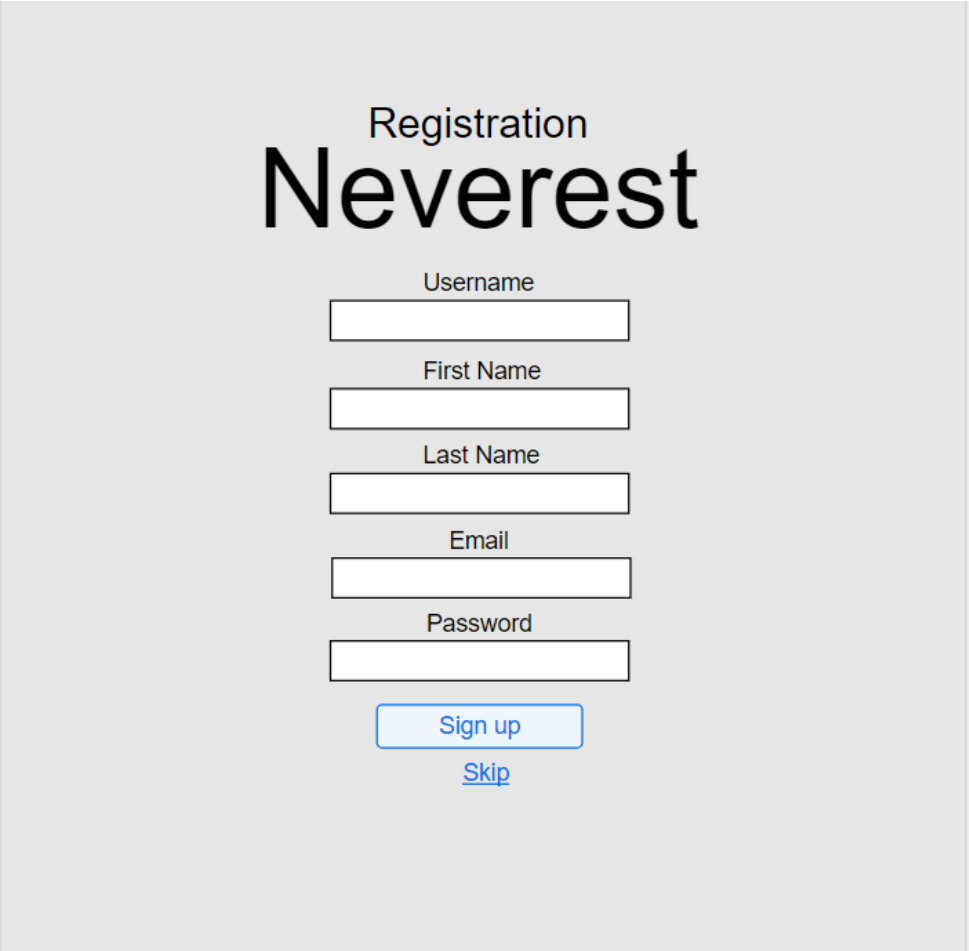
V obrazovke prihlasovania používateľ zadá svoje používateľské meno a heslo a pomocou tlačidla „Log in“ sa prihlási do systému. Nižšie pod týmto tlačidlom je dostupná registrácia nových používateľov, a taktiež preskočenie prihlásenia pre anonymných používateľov, ktorým je následne poskytnutá obmedzená funkcionalita systému.



Obrázok 15: Používateľské rozhranie

3.10.2.2 Registrácia

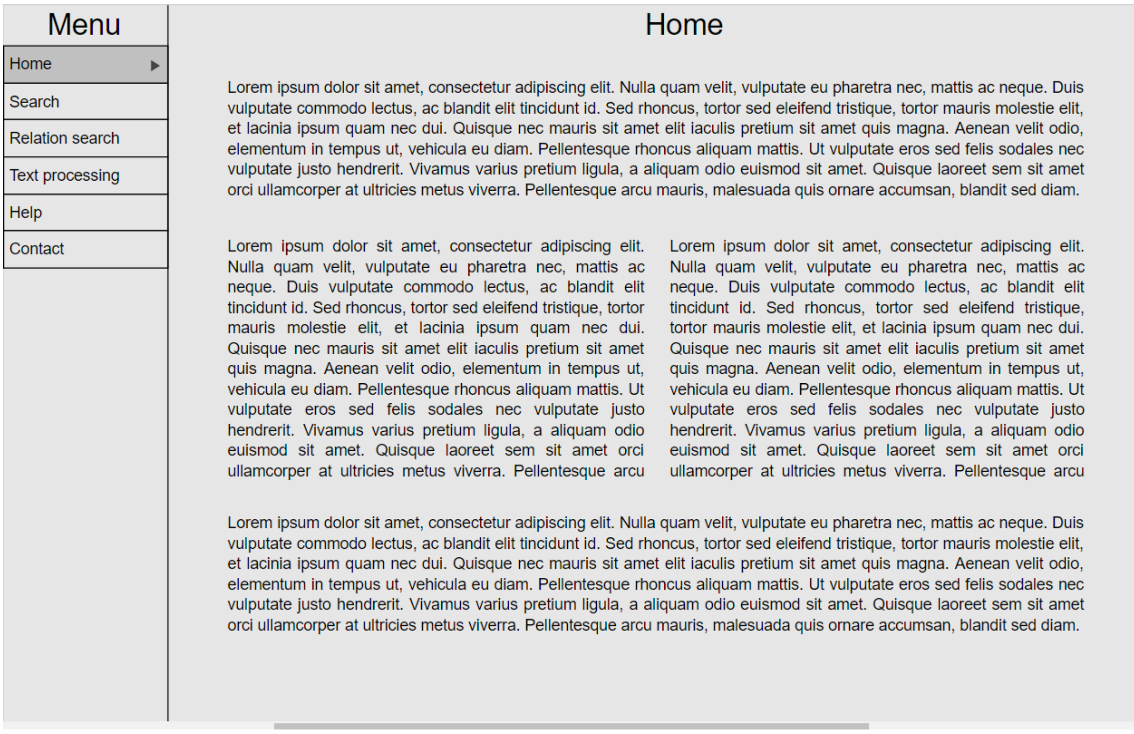
Registrácia používateľa sa vykonáva na základe zadaného používateľského mena, hesla a emailu. Používateľ do formulára vyplní tieto údaje, a odošle registráciu tlačidlom „Sign up“. V prípade, ak používateľ nemá záujem o registráciu, má možnosť ju preskočiť a pristúpiť k systému anonymne s obmedzenou funkcionalitou. Na tento úkon slúži odkaz „Skip“.



Obrázok 16: Obrazovka registrácie

3.10.2.3 Hlavná obrazovka

Hlavná obrazovka je vertikálne rozdelená na dva panely. V ľavom paneli sa nachádza menu na vyber požadovaných funkcií. Tie sa následne zobrazujú v pravom paneli.



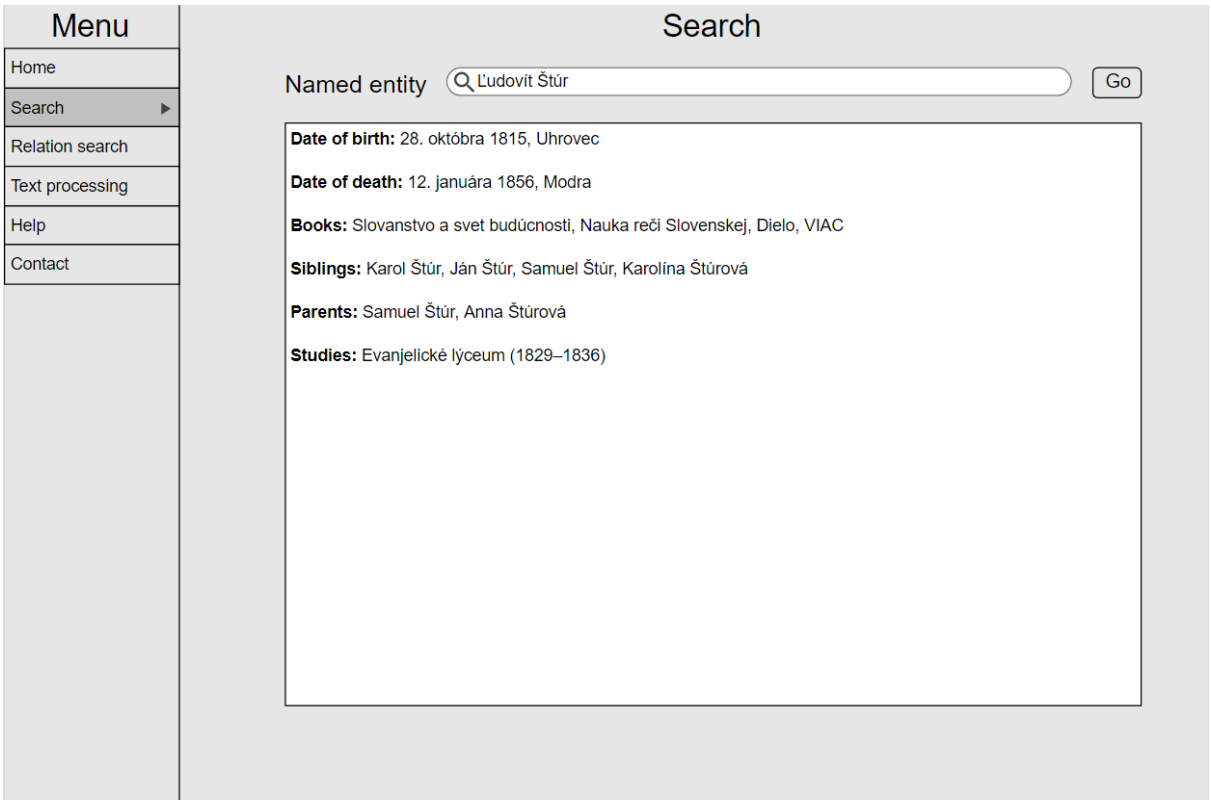
Obrázok 17: Hlavná obrazovka

3.10.2.4 Vyhľadávanie

Funkcionalita vyhľadávania je v podobe jednoduchého vyhľadávacieho poľa v databáze na základe kľúčových slov zadaných používateľom. Používateľovi sa zobrazí zoznam výsledkov, v ktorom si následne môže zobraziť detailne informácie o požadovaných entitách.

3.10.2.5 Vyhľadávanie vzťahov

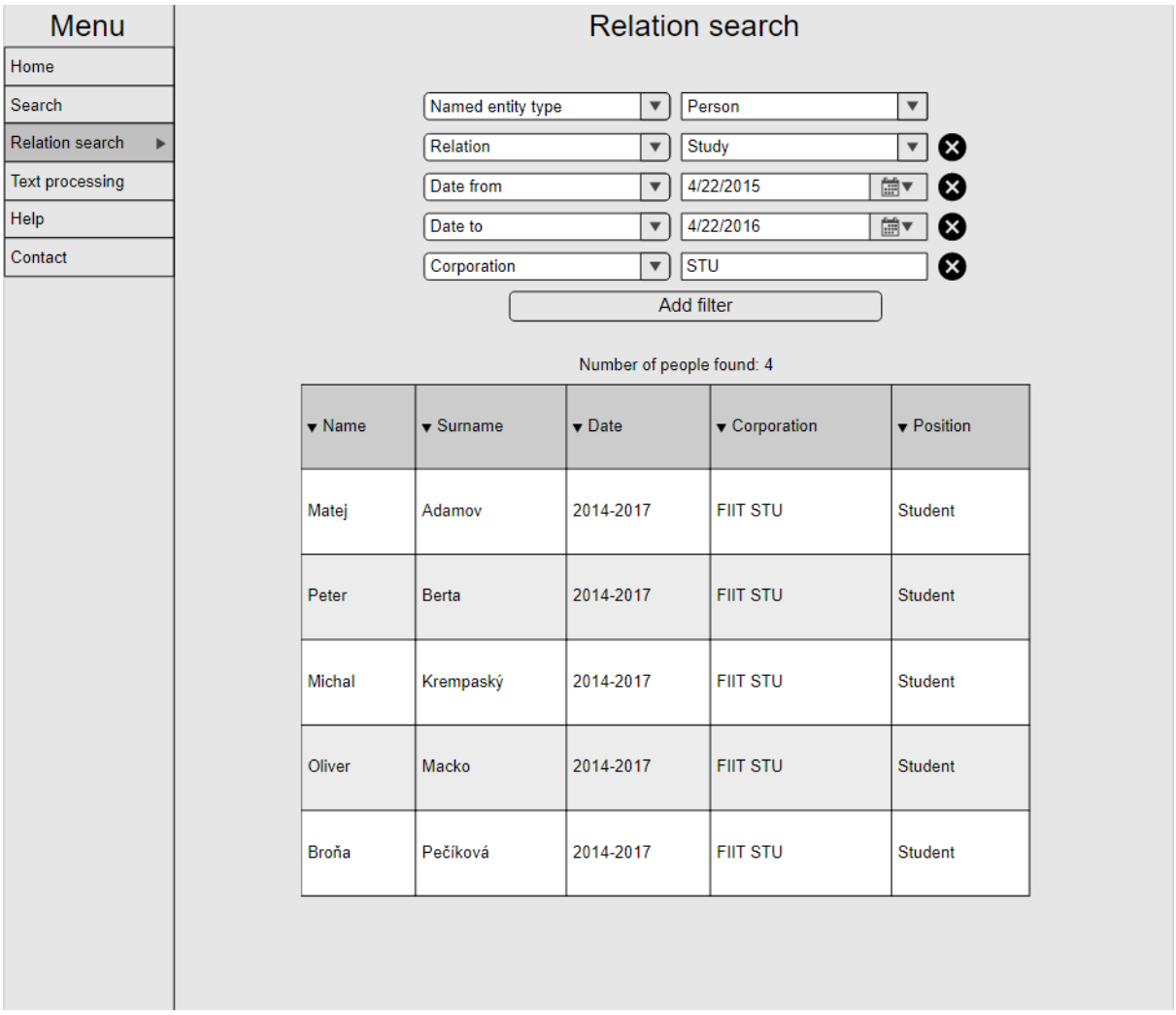
Pri vyhľadávaní vzťahov si používateľ zadáva viacero kritérií vyhľadávania. Tieto kritéria sú v požívateľskom rozhraní reprezentované filtrami, ktoré je možné pridávať a odoberať podľa potreby. V týchto filtroch používateľ vie definovať časovú os, korporáciu, miesto, vzťah, osobu, podľa toho čo vyhľadáva. Výsledkom tohto vyhľadávania je kolekcia všetkých záznamov, ktoré zodpovedajú zadaným kritériám.



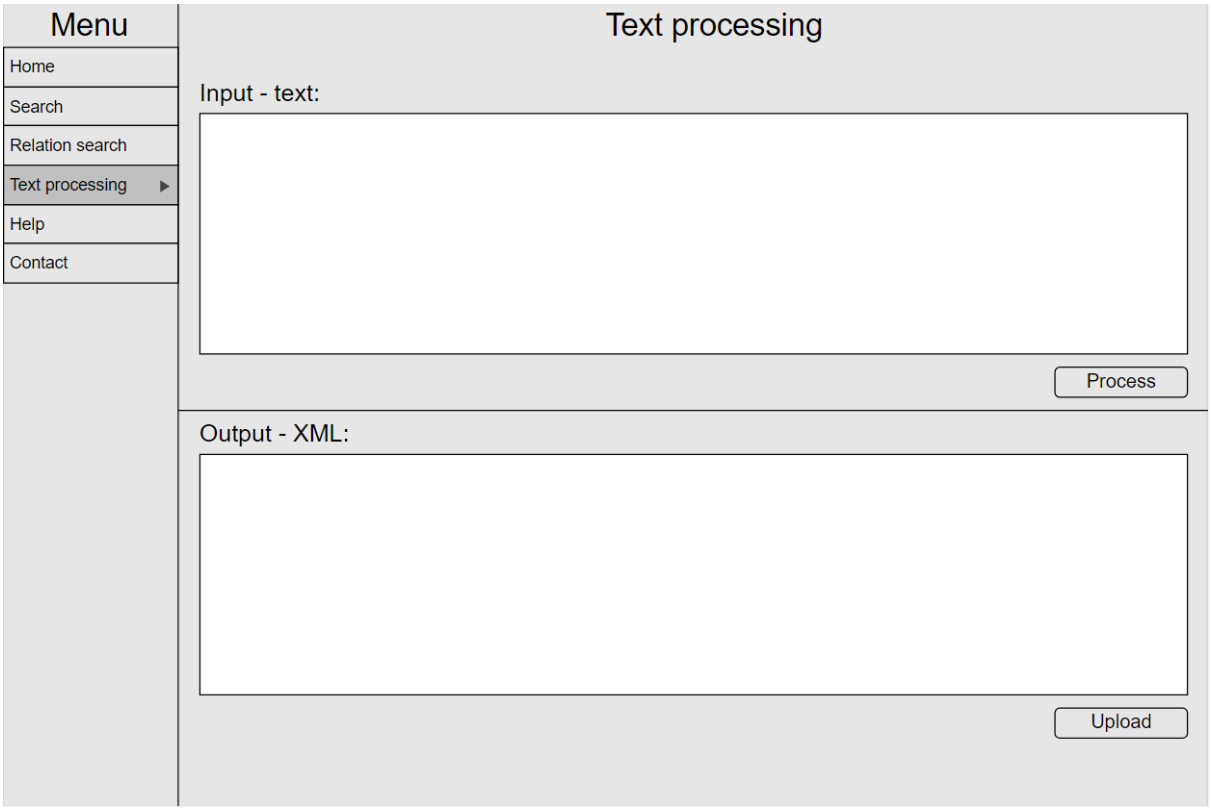
Obrázok 18: Obrazovka vyhľadávania

3.10.2.6 Spracovanie textu

Spracovanie textu je nástroj webového rozhrania, ktorý umožňuje používateľovi z neštruktúrovaného textu životopisu vytiahnuť informácie o štúdiu. Výsledné informácie sú vo formáte XML. Na spracovanie textu slúži tlačidlo „Process“. Tlačidlo „Upload“ slúži na vloženie štruktúrovaných dát do databázy. Táto možnosť je dostupná iba používateľom s potrebnými právami prístupu.



Obrázok 19: Vyhľadávanie podľa vzťahov



Obrázok 20: Obrazovka spracovania textu

3.11 Implementácia používateľského rozhrania

Na implementáciu používateľského rozhrania sme zvolili framework Django. Veľkou výhodou bolo, že naša back-end aplikácia je naprogramovaná vo forme pipeliny v jazyku Python. To nám značne uľahčilo integráciu pipeliny, v porovnaní inými frontendovými frameworkami.

Výhody:

- rýchlosť - zámer navrhnutia tohto frameworku je uľahčiť programátorom, aby aplikácie bežali plynulo a rýchlo,
- plná výbava – Djano framework poskytuje veľa doplnkov, ktoré napomáhajú s autentifikáciou používateľov, manažovanie databázových modelov a pod,
- bezpečnosť – predchádza chybám spojených s bezpečnosťou SQL databáz,
- škálovateľný – dokáže sa vysporiadať s veľkou migráciou dát,
- všestrannosť – dokáže integrovať vedecko-výpočtové platformy a tiež systémy veľkých organizácií.

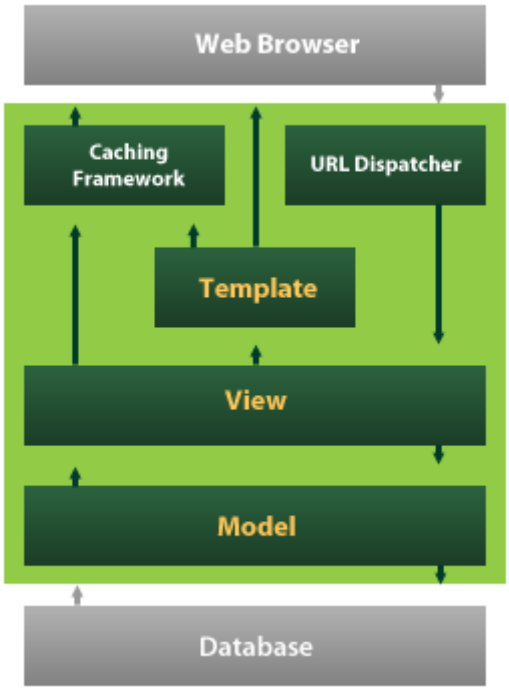
Nevýhody:

- používanie presmerovacieho systému slúžiace na presmerovanie používateľa na inú URL. Teda pre každú unikátnu URL musí existovať práve jedna URL. Problém sa javí najmä pri robustnejších aplikáciách, kde používateľ stráca prehľad o adresách zabezpečujúcich presmerovanie.

3.11.1 Architektúra frameworku Django

Django sa veľmi podobá na vzor MVC, ale na rozdiel od MVC zvykne používať vlastnú logiku v implementácií. Dôvodom je fakt, že riadenie "C - controllerom" je zabezpečené samotným frameworkom. Najviac akcií je vykonaných v modeli, šablóne (template) a view. Django často označujeme ako MTV framework (model-template-view).

- M - Model: model predstavuje dátovú vrstvu, ktorá obsahuje informácie o dátach, ako k nim pristupovať, ako ich validovať, charakterizuje ich správanie a vzťahy.
- T - Template (šablóna): prezentačná vrstva, ktorá rozhoduje o prezentačných vzťahov ako napr. ako nejaký element bude zobrazený na webovej stránke
- V - View: ide o vrstvu zabezpečujúcu biznis logiku. Vrstva obsahuje logiku, ktorá pristupuje k modelu a odkazuje sa na konkrétnu šablónu. Môžeme si túto vrstvu predstaviť tiež ako "most" prepájajúci model a šablónu.



Obrázok 21: Architektúra frameworku Django

3.11.2 Architektúra aplikácie DeepSearch

Architektúra našej aplikácie sa v porovnaní s analyzovanou architektúrou líši hlavne v tom, že chýba v nej model, ktorý je súčasťou beckendu. Webová aplikácia sa skladá z View a Template-u. Interakcia medzi View a Template-om je zabezpečená aj pomocou technológie AJAX. Volanie modelu medzi webovou aplikáciou a beckendom, ktorý je tvorený pipelinou, je zabezpečené pomocou fasády. Fasáda obsahuje inicializovanie neo4j databázy a volania metód prezentačnej vrstvy modelu na beckende.

Webová aplikácia sa skladá z týchto modulov:

- Home
- Search_entity
- Text_processing
- Graph

3.11.3 Modul home

- Vrstva - View: Vrstva view v tomto moduly je reprezentovaná triedou MultipleModelView(TemplateView), ktorá obsahuje metódu get_context_data(), ktorá vracia zoznam identifikovaných entít typov Person, City a Corporation. Zoznam je usporiadaný podľa počtu vzťahov s inými identifikovanými entitami. Pri prvotnom načítaní entít (pri nezadaní časovej periódy), sa načítajú len top 20 identifikovaných entít s najväčším počtom vzťahov bez ohľadu na časovú periódu. V opačnom prípade sa zobrazia entity v rámci vybranej časovej periódy.
- Vrstva - Template: Názov šablóny, do ktorej sa zobrazí obsah metódy get_context_data() je home.html. V šablóne pristupuje k identifikovaným entitám cez objekt slovník. To znamená, že v šablóne musíme zadať parameter "key" slovníka a potom už len pomocou objektového prístupu pristupujeme k jednotlivým atribútom a tie zobrazujeme v šablóne. Napr. person.name vypíše meno objektu (ak ide o jeden objekt).

Modul home môžeme vidieť v týchto scenároch požívateľskej príručky: Scenár 2.1 a 2.2

Modul home obsahuje len jednu URL na presmerovanie - #. Ide o prvotné načítanie entít resp. načítanie entít vybraním časovej periódy.

3.11.4 Modul search_entity

- Vrstva - View: Vrstvu view tohto modulu charakterizujú dve funkcie.

  Prvou je funkcia search_id(), ktorá je volaná pri vyhľadávaní entít na základe mena (požívateľská príručka - scenár 4.1). Funkcia spracuje vstup zo šablóny. Potom na základe zadaného mena sa nájde v modeli (na beckende prostredníctvom fasády) pomocou funkcie find_person() entita typu Person. Následne na základe mena osoby sa prostredníctvom funkcie get_person_rels() získajú všetky vzťahy, ktoré sú prepojené s vybranou entitou. V treťom kroku sa z Django modelu Document získa text životopisu patriaci vybranej entity. V poslednom kroku sa získajú informácie o narodení a úmrtí vybranej entity. Všetky tieto čiastkové dáta sa uložia do slovníka pod unikátnym kľúčom a následne slovník je dostupný po zavolaní funkcie search_id().

  Druhou je funkcia search_id_a(), ktorá má rovnakú funkcionalitu ako prvá spomínaná funkcia, ale nepoužíva sa pri hľadaní entity pomocou formuláru, ale pomocou dvojklikom na meno entity typu Person v grafe. Návratová hodnota funkcie nie je renderovanie Django obrazovky patričnou šablónou, ale vráti sa html vo forme HttpResponse. Pri použití tejto funkcie sa používa technológia AJAX, ktorá zabezpečí volanie funkcie search_id_a() pomocou JavaScriptu.

- Vrstva - Template: Názov šablóny, ktorá zobrazuje dáta týchto dvoch funkcií sa nazýva search_id.html. Šablóna obsahuje pomocný JavaScript, ktorý zahŕňa implementáciu grafovej vizualizácie pomocou knižnice vis.js (viac v moduly - graph) a tiež aj spomínane AJAX - POST a GET metódy na vyhľadávanie entít. Na obrázku č. 22 môžeme vidieť ukážku volania vrstvy - View pomocou AJAX-u.

```
$.ajax({
    type: "GET",
    url: '/search_name/'+name+'/',
    success:function(response) {
        location.assign('/search_name/'+name+'/');
        $('#xx').html(response);
    }
});
```

Obrázok 22: Ukážka AJAX volania

Modul search_entity môžeme vidieť v scenároch 3.1 - 3.4 a 4.1 požívateľskej príručky.

Modul search_entity obsahuje dve URL na presmerovanie. Prvou je vyhľadanie entity pomocou elementu form v šablóne a druhou je vyhľadanie entity pomocou grafu (dvoj-kliknutím na uzol). Zachytenie volania zabezpečuje knižnica Jquery, ktorá zaznamená akciu kliknutia na uzol v grafe a zavolá príslušnú funkciu vo vrstve View.

3.11.5 Modul text_processing

- Vrstva - view: Vrstva view v tomto module je reprezentovaná dvoma funkciami:

  Prvou je funkcia process_cv(), ktorá pomocou formuláru POST v šablóne získa meno a text životopisu. Po získaní údajov zo vstupu sa uložia do internej PostgreDB a vráti sa id (privátny kľúč), ktorý bude vstupným parametrom funkcie process_single_cvs(). Pomocou fasády sa spomenutá funkcia zavolá a vráti slovník s kľúčom "triplets". Formát slovníka s kľúčom "triplets" je JSON, ktorý sa upraví pomocou funkcie json.dumps() a následne sa slovník pošle do šablóny ako HttpResponse.

  Druhou funkciou je uloženie tripletov resp. identifikovaných entít získaných z funkcie process_cv(). Uloženie tripletov do databázy neo4j zabezpečí volanie funkcie neo_store(), ktorá získa triplety z Output okna a identifikačný kľúč bundlu. Následne sa zavolá funkcia import_relationships() s vyššie spomenutými parametrami. Návratová hodnota funkcie je premenná status, ktorá určuje, či sa triplety úspešne uložili do neo4j DB.

- Vrstva - template: Názov šablóny, ktorá zobrazuje dáta týchto dvoch funkcií sa nazýva text_processing.html. Šablóna obsahuje pomocný JavaScript, ktorý zahŕňa implementáciu volaní AJAX, ktorým vieme zabezpečiť, aby po kliknutí na tlačidlo "process_cv" sa znova nevykreslila a neobnovila stránka. Pre každú funkciu vo vrstve View je implentované POST - AJAX volanie. V prípade funkcie processCVs() sa po úspešnom volaní zobrazia triplety(formát JSON) v šablóne - Output okno. Po úspešnom volaní AJAX funkcie storeNeo() je návratovou hodnotou premenná "success", čo indikuje, že sa triplety úspešne uložili a zobrazí sa dialógové okno o úspešnom uložení. Obidva prípady je možné vidieť v požívateľskej príručke (viď Scenár č. 5.1).

Modul test_processing môžeme vidieť v scenároch 5.1 požívateľskej príručky.

Modul test_processing obsahuje dve URL na presmerovanie. Prvou je spracovanie textu životopisu a druhou je uloženie spracovaných tripletov do neo4j databázy. Zachytenie volania zabezpečuje knižnica Jquery, ktorá zaznamená akciu kliknutia na tlačidlo a zavolá príslušnú funkciu vo vrstve View.

3.11.6 Modul graf (vis.js)

Tento modul je súčasťou modulu search_entity, ale preto, že obsahuje značnú časť implementácie, sme ho zaradili medzi moduly.

Modul je zložený z týchto častí:

- statická premenná AUTHORIZATION - definuje nastavenia(meno a heslo neo4j DB), pomocou ktorých sa webový server pripojí na neo4j DB.
- funkcia restPost() - ide o AJAX volanie, v ktorom sa určuje typ AJAX volania, hlavička requestu, URL na prepojenie s neo4j DB, návratový typ a premennú, do ktorej sa uložia dáta.
- funkcia displayGraph() - ide o hlavnú funkciu, ktorá obsahuje v sebe nastavenia autorizácie a dopytu, pomocou ktorej získame dáta. Po úspešnom volaní sa tieto dáta (objekty) pomocou funkcií parseGraphResultData(), convertNodes() a convertEdges() rozložia na uzly a hrany grafu. Následne sa zavolá funkcia displayVisJsData(), ktorá tieto uzly a hrany zobrazí.
- funkcia displayVisJsData() - pozostáva z 3 častí. Prvou je nastavenie kontajneru tj. elementu do ktorého sa vykreslí graf. Druhým je definovanie nastavení vizuálnych aspektov jednotlivých elementov (farba uzlov, hrán, typ písma a pod.) a treťou časťou inicializácia grafu (viď obrázok 23), ktorá môže obsahovať event akcie (mouse hover, popUp okno, click event, doubleClick event a pod.)

```
// initialize the network!
var network = new vis.Network(container, data, options);
```

Obrázok 23: Ukážka inicializácie grafu pomocou knižnice vis.js

- funkcia parseGraphResultData() - zabezpečí získanie objektov z neo4j DB a ich transformáciu
- funkcie: convertNodes() a convertEdges() - zabezpečujú, ktoré atribúty uzlov resp. hrán sa vykreslia v grafe