Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Porovnání obsahu výuky oborů na univerzitách

Plzeň 2014 Bc. Pavel Slavíček

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů. Dále poskytuji celý obsah této diplomové práce k účelům Západočeské univerzity.

V Plzni dne 13. května 2014

Bc. Pavel Slavíček

Poděkování

Děkuji vedoucímu práce Prof. Ing. Karlu Ježkovi CSc. za odborné vedení a cenné rady při vypracování diplomové práce. Dále děkuji společnosti Owen Software Development Compeny Ltd. za poskytnutá data, bez kterých by tato diplomová práce nemohla vzniknout. Poděkovat bych také chtěl mým nejbližším, kteří mě po dobu celého studia podporovali.

Abstract

The aim of this thesis was to explore the existing methods and tools which enable searching through data. Furthermore, the aim was to apply the acquired knowledge in developing of a system focussed on searching of courses which are offered by particular universities. The second part of the thesis deals with description of existing means needed for the realization of the (newly) designed system. This part also contains substantiation of their choice in comparison with other alternatives. The next part of the thesis describes the designed strategies, which are necessary for reaching of the demanded features of the created system. Finally, the thesis gives the description and evaluation of the results in conjunction with the created functions.

Obsah

1 Úvod

Podnětem k zadání této diplomové práci byl požadavek vytvořit takový systém, který by měl budoucím studentům usnadnit výběr univerzity. Tento systém by měl poskytnout zájemcům o studium v přehledné formě co nejvíce informací o nabízených kurzech z celé řady (možných) univerzit. S pomocí takového systému bude zájemce mít větší možnosti porovnání celého množství kurzů a tak se může snadněji rozhodnout pro výběr pro něj vhodné univerzity. Tato služba bude využívat databázi kurzů propůjčenou společností *Owen Software Development Compeny Ltd.* Od aplikace je požadováno, aby měla formu webové služby.

Během vyhledávání kurzů je požadováno, aby systém uživateli napomáhal. Výpomoc uživateli je možná formou odhadu jeho zájmů o určitý obor na základě hledaných kritérií. Kromě odhadu jeho zájmů se nabízí také jistá forma nápovědy pomocí alternativních slovních vyjádření k hledaným kritériím. Tato forma nápovědy má uživateli umožnit vyhledávat i bez znalosti konkrétních pojmů z daného oboru.

Aplikace bude poskytovat funkce pro zpracování zdrojových dat a bude možné nastavit cesty ke zdrojovým souborům s daty. Dále aplikace bude mít funkce pro vyhledávání v kurzech, které budou využívat prostředky a navrhnuté strategie za účelem usnadnění vyhledávání uživateli.

2 Popis struktury dat

Pro účely této diplomové práce byla poskytnuta struktura dat ve formě třiceti souborů formátu XML s názvy `allcoursesPORADOVECISLO.xml`. O těchto XML souborech bude v této práci pojednáváno zároveň i jako o databázi dat, XML databázi nebo jen databázi, i když v současnosti pojem databáze zahrnuje také softwarové prostředky umožňující manipulaci s daty. Obecně je databáze jakákoliv uspořádaná množina informací uložená na paměťovém médiu.

Poskytnutá databáze obsahuje informace o nabízených kurzech na univerzitách ve Spojených státech amerických. Informace o kurzech obsažené v sadě XML dokumentů jsou uloženy v anglickém jazyce. Struktura XML postrádá určení znakové sady. Znaková sada u všech souborů byla zjištěna unixovým programem *file*. V kolekci XML dokumentů existují dvě různé znakové sady, kterými jsou UTF-8 a US-ASCII. Soubor s kódováním UTF-8 je totožný se znakovou sadou US-ASCII, pokud obsahuje jen znaky z anglické abecedy.

Pro daná data byla za pomoci nástroje Oxygen XML Editor[1] vygenerována definice XML struktury formou XSD schématu pro všechny jednotlivé XML soubory. Všechny definice k XML dokumentům jsou na přiloženém CD k této práci.

2.1 Struktura dat

Všechny dokumenty XML jsou správně strukturované tzv. „well-formed“[2] a jejich struktura je navzájem shodná. Pro všechny dokumenty platí, že kořenový element je *courses*, který obsahuje data jednotlivých kurzů. Každý kurz je uzavřen v elementu *course* a je v něm vnořeno několik dalších elementů, které blíže specifikují daný kurz. Všechny elementy kromě kořenového a *course* tj. elementy, které blíže specifikují informace o kurzu, dále elementy *token* jehož vysvětlení se nachází pod následující ukázkou popisované XML struktury.

[1] Oxygen XML Editor je multiplatformní XML editor, XSLT/XQuery nástroj pro ladění programu. Více informací je na domovské stránce nástroje tj. na adrese `http://www.oxygenxml.com/`.

[2] Vysvětlení významu označení dokumentu jako *well-formed* se nachází na stránkách `http://en.wikipedia.org/wiki/Well-formed_document`

```
<courses>
  <course luceneId='1'>
    <courseNumber type='java.lang.String'>
      <token word='47777' tag='CD' chunkTag='O' />
    </courseNumber>
    <institution type='java.lang.String'>
      <token word='University' tag='NNP' chunkTag='B-NP' />
      <token word='of' tag='IN' chunkTag='B-PP' />
      <token word='North' tag='NNP' chunkTag='B-NP' />
      <token word='Texas' tag='NPP' chunkTag='E-NP' />
    </institution>
    <yearOffered type='java.lang.Integer'>2010</yearOffered>
  </course>
<courses>
```

Element *token*

Atribut *word* elementu *token* obsahuje jedno slovo nebo takový textový řetězec, který je považován za jeden token. Další atribut *wnGloss* popisuje sémantiku obsahu atributu *word*. Tento element má ještě několik atributů, u kterých však nelze s určitostí říci, jaký mají význam. Neurčitost významu je dána zaprvé stručností názvu atributu a především jeho obsahem a za druhé faktem, že nebyl k databázi dodán žádný upřesňující popis. Ovšem názvy u elementů jsou ve všech případech samopopisující. Nutno dodat, že některé atributy nejsou povinné. Ukázka jednoho elementu *token* je uvedena níže.

```
<token word='Selected' tag='VBB' chunkTag='B-PP' wnSense='1'
    wnPos='v' wnGloss='pick␣out,␣select,␣or␣choose␣from␣a␣
    number␣of␣alternatives' />
```

2.2 Transformace dat

Z důvodu velikosti databáze, kterou zabírá na paměťovém médiu bylo žádoucí pro další manipulaci transformovat data do jiné podoby XML[3]. Transformace dat je realizovatelná z důvodů nadbytečnosti některých informací v databázi, které nejsou pro účely této práce podstatné. Tyto nadbytečné informace se odstraní při převodu databáze do jiné.

[3]Jakákoliv manipulace s daty se uskutečnila pomocí SAXON XSLT procesoru (`http://saxon.sourceforge.net/`) a XSLT schémat

Pro účel transformace dat bylo vytvořeno XSLT schéma[4]. Podle vytvořeného XSLT schématu se převedou zdrojová data do téměř stejné XML struktury s tím rozdílem, že se všechny elementy *token* nahradí obsahem atributu *word* téhož elementu. Další a poslední rozdíl je ve vynechání elementu `originalID`, protože se jedná o složeninu dvou jiných elementů. Element `originalID` se skládá z elementů `deparment` a `courseNumber`. Tento převod zmenšil původní data v průměru o 85%. Obsahy nových souborů lze sloučit do jednoho souboru podle XSLT schématu[5] vytvořeného speciálně pro tento účel. Pro převedená data byla vygenerována definice dat XSD pomocí nástroje Oxygen XML Editor.

Níže je uvedena ukázka původního zdroje dat, za níž následuje další.

```
1  <courses>
2    <course luceneId='1'>
3      <courseNumber type='java.lang.String'>
4        <token word='47777' tag='CD' chunkTag='O' />
5      </courseNumber>
6      <institution type='java.lang.String'>
7        <token word='University' tag='NNP' chunkTag='B-NP' />
8        <token word='of' tag='IN' chunkTag='B-PP' />
9        <token word='North' tag='NNP' chunkTag='B-NP' />
10       <token word='Texas' tag='NPP' chunkTag='E-NP' />
11     </institution>
12     <yearOffered type='java.lang.Integer'>2010</yearOffered>
13   </course>
14 <courses>
```

Následující ukázka odpovídá výsledku transformace první ukázky uvedené výše.

```
1  <courses>
2    <course luceneId='1'>
3      <courseNumber type='java.lang.String'>
4        47777
5      </courseNumber>
6      <institution type='java.lang.String'>
7        University of North Texas
8      </institution>
9      <yearOffered type='java.lang.Integer'>2010</yearOffered>
10   </course>
11 <courses>
```

[4]XSLT schéma pro převod originálního XML souboru do redukovaného souboru se jmenuje `allcourses_to_reduced.xsl` viz příloha A.1

[5]XSLT schéma pro sloučení všech třiceti XML souborů má název `all_merge_allcourses_reduced.xsl` viz příloha A.1

2.3 Statistiky

V rámci studie dat byly vytvořeny statistky popisující upravenou databázi, jejíž transformace z původních dat je popsána v sekci 2.2. Statistiky o celkovém počtu kurzů, kateder a univerzit jsou uvedeny v tabulce 2.1.

Popis	**Počet**
Univerzit	29
Keteder	3201
Kurzů	299519

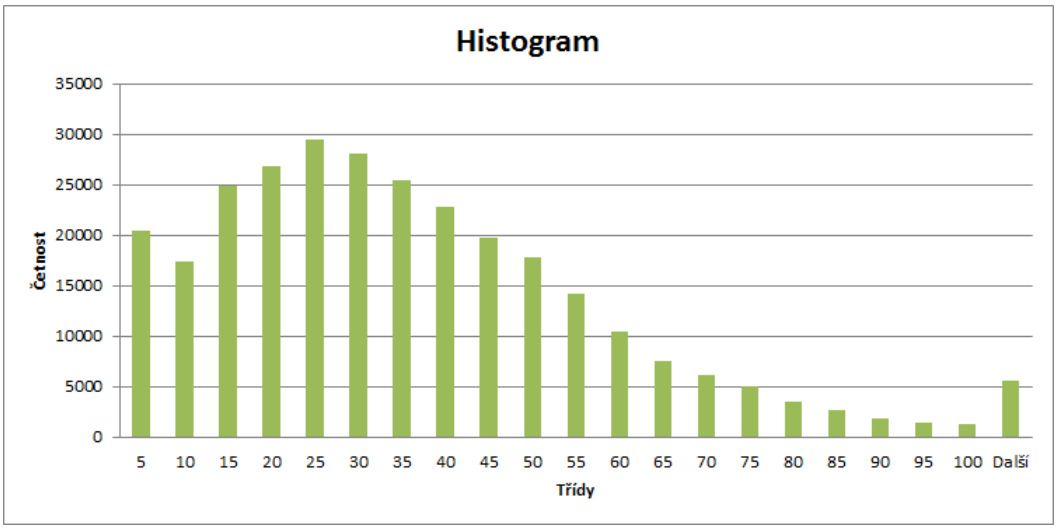
Tabulka 2.1: Počet univerzit, kateder a kurzů v databázi.

Nutno poznamenat, že se v databázi nenacházejí kurzy příslušící například univerzitám: *Massachusetts Institute of Technology*, *Stanford University*, *Carnegie Mellon University*, *Harvard University* atd. Vyjmenované univerzity jsou umístěny na předních příčkách podle hodnocení kvality v oblasti výpočetních věd uvedené v „QS World University Rankings by Subject 2013 - Computer Science & Information Systems“[6].

Element *overview*

Relativně důležitým elementem v celé databázi je element *overview*. Tento element obsahuje popis daného kurzu, který lze významem přirovnat k sylabu. V následujícím obr. 2.1 je uveden histogram četnosti slov v tomto elementu, přičemž jednotlivá slova byla detekována na základě mezer v textu. Zvolená technika určování počtu slov v elementu *overview*, ačkoliv přímočará, je pro účely popisu dat zcela dostačující.

[6]Domovské stránky organizace provádějící toto hodnocení jsou `http://www.topuniversities.com/`



Obrázek 2.1: Histogram počtu slov v elementu *overview*.

Elementy obecně

Celkový počet možných elementů v jednom kurzu je 34. Bylo zjištěno, že jen elementy *institution* a *yearOffered* se pokaždé vyskytují v kurzu a ostatní elementy nejsou povinné. V tabulce 2.2 jsou uvedeny nejpočetnější a „nejdůležitější“ elementy. Tabulky a grafy, které uvádějí konkrétní počty výskytů a vysvětlující popis u všech elementů je možné zhlédnout v příloze A.3.

Element	Počet výskytů
institution	299519
yearOffered	299519
courseNumber	299512
department	299512
title	296791
overview	292594

Tabulka 2.2: Seznam elementů s uvedeným počtem výskytů.

3 Použité prostředky

V této kapitole je uveden popis několika prostředků nutných pro realizaci aplikace vytvořené v rámci této diplomové práce. Prvním popisovaným nástrojem je lemmatizátor, jenž poskytuje také jiné funkce. Druhým zvoleným prostředkem je tezaurus použitý k získávání množiny podobných výrazů k danému pojmu a obdobně jako tezaurus je použit i ACM klasifikační systém. Smyslem zvolených prostředků, konkrétně lemmatizátoru, tezauru a ACM klasifikačního systému je rozšiřování dotazu zadaného uživatelem. Tazatel totiž pravděpodobně nezná nebo nechce doslovně zadávat všechny používané termíny typické pro danou oblast, o kterou se aktuálně zajímá. Toto rozšíření dovoluje zadat tazateli například jen slovo „network“ a v případě kdy se ve zdrojích použitých prostředků najdou relevantní pojmy, jsou použity k rozšíření dotazu. Popis konkrétního způsobu použití všech prostředků v aplikaci je uveden v kapitole 4.

Seznam *stop list* je použit podobným způsobem jako ostatní prostředky tzn. je součástí procesu rozšiřování dotazu. Stop list je jednoduchý seznam slov, která mají v textu malou vypovídající hodnotu. Jedná se zejména o slova, jako jsou zájmena, předložky atd. Zvolený stoplist je určen pro anglický jazyk a je uložen ve formě textového souboru s jednotlivými slovy na každé řádce. Seznam stop list je dostupný na přiloženém CD k této práci viz příloha A.1.

3.1 Lemmatizátor

Jak je popsáno v [Michal(2006)] provádí tento nástroj lemmatizaci textu, což je v lingvistice proces převedení různých tvarů zkoumaného slova na základní tvar tzv. lemmu, která je často vhodnější pro další zpracování textu. Lemmatizace může být například součástí procesu porovnávání dvou textů na jejich podobnost. Náročnost procesu lemmatizace se odvíjí od komplexnosti jeho implementace a v závislosti na přirozeném jazyku, pro který je určen. Některé pokročilé lemmatizátory mohou analyzovat také celé věty k určení správného tvaru. V souvislosti s lemmatizátory se také diskutuje o tzv. stemmatizátorech, které ze zkoumaného slova extrahují kmen slova.

Existuje několik implementací lemmatizátoru. Některé implementace jsou

komerční a další jsou volně šiřitelná. Zvoleným nástrojem pro účely aplikace v této práci je volně šiřitelný *Stanford CoreNLP*[1], který mimo jiných funkcí umožňuje také lemmatizaci. Stanford CoreNLP je komplexní nástroj určený pro „the part-of-speech (POS) tagger“, „the named entity recognizer (NER)“, „the parser“, „the coreference resolution system“, „the sentiment analysis tools“ a poskytuje tzv. model vzorových souborů pro analýzu angličtiny. Volba tohoto lemmatizátoru je opodstatněná jeho dobrou dostupností a také tím, že poskytuje požadované vlastnosti. Požadovanými vlastnostmi jsou především jeho zaměření na anglický jazyk a implementace v programovém jazyce Java. Existuje celá řada alternativ ke zvolenému lemmatizátoru, ale většina z nich nejsou volně šiřitelné pro nekomerční účely.

3.2 Tezaurus

Tezaurus je takový slovník, který zahrnuje především seznam synonym a v některých případech také seznam antonym. Tezaurus ve spojitosti s vyhledáváním informací může být použit jako řízený slovník deskriptorů, mezi kterými jsou definovány vztahy nadřazenosti a podřazenosti, synonymní a jiné související termíny. Tento slovník nabízí shodné nebo podobné seznamy slov, které popisují stejnou problematiku. Použití tohoto nástroje v aplikaci umožňuje obsáhnout větší množinu relevantních témat bez znalosti přesného vyjádření problematiky anebo naopak vyznat se v nepřeberném množství informací. Poslouží tedy jako propojovací jazyk v informačních systémech. Další informace o tezauru lze nalézt v [Michal(2006)].

WordNet

Zvoleným nástrojem plnícím roli tezauru je *WordNet*[2]. Podle [Wor(2014)] je WordNet rozměrná lexikální databáze anglického jazyka. Podstatná jména, slovesa, přídavná jména a příslovce jsou sjednocovány do souborů kognitivních synonym (synsets), z nichž každý vyjadřuje odlišnou koncepci. WordNet se podobá tezauru v tom, že seskupuje slova podobného významu do jednot-

[1] Nástroj *Stanford CoreNLP* je dostupný na stránkách `http://nlp.stanford.edu/software/corenlp.shtml`

[2] WordNet je nástroj vyvíjen na univerzitě Princeton a je dostupný na stránkách `http://wordnet.princeton.edu/wordnet/`, kde je také možné nalézt více informací a návod na instalaci.

livých kategorií. WordNet má několik odlišností od „obyčejného“ tezauru. Zaprvé vzájemně propojuje nejen slova (jako prosté posloupnosti písmen), ale také konkrétní význam slov. Výsledky nalezení k sobě blízkých slov jsou v síti WordNet sémanticky jednoznačné. Dalším rozdílem jsou popisy sémantických vztahů mezi slovy, zatímco seskupení slov v tezauru se neřídí žádným jiným explicitním vzorcem než podobností významu slov.

Popis struktury WordNet je následující. Nejdůležitějším vztahem mezi slovy je synonymie, jako například mezi anglickými slovy *car* a *automobile*. Synonyma, která označují stejné pojetí a jsou zaměnitelná v různém kontextu jsou seskupena do neuspořádaných sad (synsets). Každý synset je spojen s jiným synsetem prostřednictvím omezeného počtu tzv. „koncepčních vztahů“. Sady neboli synsety obsahují stručnou definici („gloss“) a také velmi často obsahují ukázku použití jednotlivých prvků ze synsetu. Jeden ze způsobů použití je popsán v sekci 4.2. Pro používání nástroje WordNet v programovém jazyce Java je potřeba nejprve nainstalovat a nastavit databázi. Návod instalace je dostupný na domovských stránkách nástroje[3].

Základním stavebním kamenem struktury je onen zmíněný synset. Množinu synsetů lze získat metodou, jejíž ukázka je níže (veškeré ukázky kódu jsou v programovém jazyce Java).

```
Synset[] synsets = database.getSynsets(word, SynsetType.NOUN);
```

Prvním parametrem metody je slovo, na základě kterého se specifikují synsety. Druhý parametr metody definuje jaký typ synsetů se získá, přičemž při vynechání tohoto parametru se získají všechny typy synsetů. Následně lze z jednotlivých synsetů získávat různé informace skrze metody z rozhraní `Synset` z knihovny WordNet. Příkladem některých metod jsou:

- metoda `getWordForms()` zavolaná nad objektem reprezentujícím synset vrací pole příbuzných slov ke slovu původnímu,
- pomocí metody `getUsageExamples()` se získají příklady použití slovních pojmů,
- metoda `getDefinition()` vrátí definici daného synsetu.

[3]Přímá adresa vedoucí k popisu nastavení WrodNet je `http://lyle.smu.edu/ tspell/jaws/`.

Některými odvozenými rozhraními od `Synset` jsou `AdverbSynset`, `NounSynset` a `VerbSynset`. Každá implementace poskytuje některé metody navíc od základního rozhraní. Například metodu `getHyponyms()` k získání hyponym definuje rozhraní `NounSynset`. Pro více informací je vhodné nahlédnout do dokumentací na domovských stránkách WordNet. Odůvodnění volby WordNet je stejné jako v případě lemmatizátoru, viz sekce 3.1.

3.3 ACM klasifikační systém

V této sekci bude stručně popsán ACM klasifikační systém počítačových věd (The ACM Computing Classification System - CCS[4]), dále kritéria výběru a alternativy k tomuto systému, viz odstavec „Alternativy“.

Podle [Acm(2014)] je systém ACM CCS rozsahem srovnatelný, například s Mathematics Subject Classification. Verze ACM klasifikačního systému z roku 2012 je dostupná ve třech zdrojích, kterými jsou: soubor ve formátu HTML; Microsoft Word dokument a XML dokument. ACM CCS je zaměřen na počítačové vědy a jeho struktura obsahuje pojmy a jejich podmnožiny neboli hyponyma. Bylo zjištěno, že maximální počet úrovní vnoření je šest. Důvodem zvolení tohoto klasifikačního systému z několika variant je vyhovět stanoveným kritériím, viz níže.

Kritéria výběru

Na základě charakteru začlenění prostředku do aplikace byly stanoveny základní kritéria výběru daného prostředku pro účel rozšíření dotazu. Základním požadavkem je, aby počet nalezených hyponym k hledanému pojmu musel být v průměru menší než deset. Nutno dodat, že počet nalezených hyponym nebo synonym se odvíjí jak od zvoleného prostředku tak i od způsobu jeho použití. U některých alternativ k ACM CCS by bylo relativně složité získávání relevantních pojmů, z čehož plyne druhé kritérium - „jednoduchá“ implementace. Ze základního požadavku pak přímo souvisejí i ostatní tzn. nesmí být příliš obecný ani příliš podrobný, jelikož by nalezených hyponym bylo hodně (tzn. čítající desítky pojmů) anebo velmi málo (žádný nebo jeden).

[4]Odkaz na ACM klasifikační systém je `http://dl.acm.org/ccs_flat.cfm`.

Alternativy

Alternativy k ACM CCS mohou být více či méně komplexní, například taxonomie, ontologie, různé slovníky apod. ve smyslu počítačových věd. Jediným požadavkem je, aby bylo možné získat příbuzné (hyponymní, synonymní) pojmy k původnímu výrazu a kritéria popisované výše. Jako možné alternativy místo zvoleného ACM CCS byly zkoumány:

- **List of academic disciplines and sub-disciplines**[5] - Tato alternativa není příliš vhodná, avšak je reálné ji použít. Hlavní důvod k tomuto závěru je příliš rozsáhlé členění struktury. Například získání hyponym ke slovu *Economics* by čítalo přesně 51 výsledků. Takovéto množství přesahuje zvolené požadavky viz úvod této kapitoly.
- **Library of Congress Classification Outline**[6] - U této alternativy je problém její rozsáhlost a obecnost. Jejím použitím by často docházelo k rozšíření dotazu mimo oblast zájmu v závislosti na konkrétním znění původního dotazu.
- **TAXONOMY OF FIELDS AND THEIR SUBFIELDS**[7] - Nejvhodnější alternativou je volba této taxonomie, pokud by nebyla zvolena ACM CCS. Také by bylo ovšem přípustné použití více alternativ zároveň, což by mohlo být jako další rozšíření této práce.
- **Medical Subject Headings (MeSH)** - MeSH je popisován jako komplexní řízený slovník biomedicínských pojmů, který slouží k účelům indexování knih apod., přičemž může sloužit rovněž i jako tezaurus, což by bylo pro účely této práce také vhodné. Tato verze má také vícejazyčné varianty. Existuje varianta i pro český jazyk, se kterou pracuje systém MEDVIK[8].

[5] Odkaz na stránky je `http://en.wikipedia.org/wiki/List_of_academic_disciplines`.

[6] Tento odkaz `http://www.loc.gov/aba/cataloging/classification/lcco/lcco_l.pdf` vede přímo k dané klasifikaci.

[7] Tento odkaz `http://sites.nationalacademies.org/PGA/Resdoc/PGA_044522` vede přímo k taxonomii.

[8] `http://www.nlk.cz/medvik/o-systemu-medvik`

Zdroj dat

I když ACM CCS je dostupný hned v několika formátech byl vytvořen nový dokument se stejnou informační hodnotou, ale zbavený několika nadbytečných informací. Nový dokument je ve formátu XML s elementy *ul* a *li*[9]. Tento XML dokument se nachází na přiloženém CD pod názvem *acm_ccs.xml*.

[9]Význam zvolených elementů *ul* a *li* se významem shoduje se stejnojmenými HTML tagy.

3.4 Apache Lucene

Apache Lucene je zabudován do výsledné aplikace, která vznikla v rámci této diplomové práce. Nicméně tvoří jen část z celkové aplikace a z tohoto důvodu je mu věnována jedna sekce. Výběr Lucene proběhl na základě zkušeností z oborového projektu a byla vybrána verze 4.5. V následujícím textu je uveden základní popis Apache Lucene a v omezeném rozsahu také popis funkcí vybraných převážně pro potřeby vytvořené aplikace. Další informace o knihovně Lucene jsou k nalezení v literatuře [Michael McCandless(2010)].

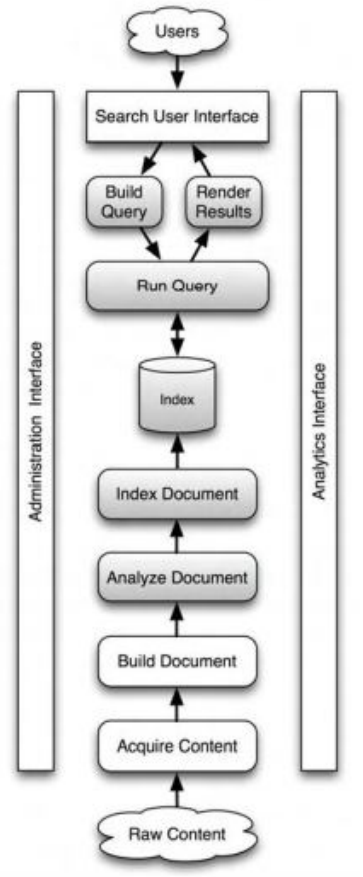
3.4.1 Popis knihovny Lucene

Apache Lucene[10] je knihovna s licencí open-source a je určena pro fulltextextové vyhledávání. Tato knihovna je kompatibilní s více operačními systémy a implementována především v programovacím jazyce Java, ale byla také portována i do programovacích jazyků Delphi (MUTIS), Perl (KinoSearch, Plucene), Objective-C (LuceneKit), C++ (CLucene), Python (PyLucene), Ruby (Ferret), PHP (Zend Search) a další. Apache Lucene je vyhledávací stroj tzv. *search engine*, který nabízí prostředky pro indexování a vyhledávání. Nabízí nízkoúrovňové API a je často výchozím bodem pro rozsáhlejší vyhledávací systémy. Jádro této knihovny zahrnuje logiku určenou pro vyhledávání v indexu, vytváření a manipulaci s indexem. Lucene dokáže indexovat téměř jakýkoliv zdroj dat, který lze převést do textové podoby.

Na následujícím obr. 3.1[11] je znázorněno, jak zhruba Lucene zapadá do vyhledávajících aplikací. Vystínované komponenty na zmiňovaném obrázku ukazují části, které Lucene zpracovává. V literatuře [Michael McCandless(2010)] v sekci 1.3 začíná podrobný popis architektury „typické“ moderní vyhledávací aplikace.

[10] Domovská stránka Apache Lucene je `https://lucene.apache.org/`

[11] Obrázek je převzat z knihy [Michael McCandless(2010)]



Obrázek 3.1: Typické složení vyhledávací aplikace.

3.4.2 Indexování

Indexování je prvotní fáze všech vyhledávacích systémů. Indexovat lze data získaná v textové podobě ze souborů uložených v různých formátech - PDF dokumenty, Microsoft Word dokumenty, XML dokumenty a jiné. Prvním krokem je tedy získání potřebných informací ze zdrojového dokumentu, které se mají indexovat. V některých případech je potřeba extrahovat obsah ze souborů, které obsahují metainformace, což Lucene sám o sobě nepodporuje a je nutné použití dalších nástrojů. V následujícím textu jsou pojmy „dokument“ a „pole“ používány v kontextu Lucene.

Dalším krokem je *sestavení dokumentů* z obsahu. Obsah, který má být indexován se musí rozdělit do jednotlivých částí nazývaných v knihovně Lucene jako dokumenty (*documents*). Tyto dokumenty jsou pak využívány při vyhle-

dávání v indexu. Jednotlivé části zmíněných dokumentů se skládají alespoň z jednoho tzv. pole, ale často jsou sestaveny z několika samostatně pojmenovaných polí. Standardně se jednotlivá pole třídí podle významu dané části textu v obsahu. Roztřídění polí může být například podle kapitol, nadpisů, autorů atd. Pole hrají důležitou roli, neboť se podle nich vyhledává a následné sloučení výsledků hledání z jednotlivých polích určují celkovou relevanci dokumentu. Celkové ohodnocení dokumentu se označuje jako skóre, které se určí na základě výsledků hledání. Protože pole mají velký význam při určování výsledného skóre, je jim umožněno přiřazovat různou hodnotu důležitosti. Ohodnocení důležitosti pole je označováno jako „boosting“ a toto ohodnocení je možné provádět staticky ve fázi indexování nebo dynamicky během vyhledávání.

Text obsažen v polích nemůže být jednoduše indexován. Před započetím indexace je textový obsah pole převeden na řetězec tzv. tokenů. Rozdělení do tokenů se provádí podle jistých pravidel, která jsou definována v použitém analyzátoru. Lucene poskytuje hned několik hotových analyzátorů a také umožňuje vytvořit si i vlastní pro konkrétní potřeby. Posledním krokem je samotná indexace dokumentu. Indexování s Lucene lze tedy shrnout do třech hlavních operací: extrahování textu z kolekce dokumentů, analyzování dat a uložení výsledku analýzy do indexu.

3.4.3 Proces analýzy při indexaci

Analyzování obsahu se provádí přidáváním dokumentů do instance třídy `IndexWriter` knihovny Lucene. Přidávání dokumentů se započne zavoláním metody `addDocument(Document)` třídy `IndexWriter`. Jak je zmíněno výše objekt `Dokument` se skládá z několika objektů `Field` tedy z polí. Každá instance třídy `Field` má své pojmenování, obsah předaný při vytváření objektu a nastavenou volbu jakým způsobem Lucene bude indexovat informace uložené v poli. V níže uvedeném textu je popsáno několik vybraných vlastností třídy `Field`.

Třída `Field`

Při vytvoření instance třídy `Field` je možné předat konstruktoru název, hodnotu a nastavit typ. Názvem se rozumí identifikace pole, hodnotou je ukládaný text a typem je například *double*. Nicméně v novějších verzích (v této

práci se diskutuje o Lucene verze 4.5) knihovny Lucene se dají vytvářet pole pomocí oddděných tříd od `Field`. Následuje výčet vybraných tříd oddděných od třídy `Field`.

- **TextField** - Tato třída je vhodná pro obsah, který má být uchován v původní podobě pro pozdější zpracování. Používá se pro obsah s velkou mírou důležitosti. Obsah se indexuje a tokenizuje.
- **StringField** - Obsah je indexován, ale nikoliv tokenizován. Vhodné pro obsah nesoucí například hodnotu ID.
- **IntField** - Pole, které indexuje číselné hodnoty (datového typu integer) pro zefektivnění filtrování a řazení podle čísel.

Obecně všechny typy polí umožňují za využití výčtového typu `Field.Store` nastavit, zda se původní obsah uloží do pole.

3.4.4 Vyhledávání

Po vytvoření indexu je možné přikročit k fázi vyhledávání. Zadaný požadavek na vyhledání je převeden do odpovídajícího dotazu přizpůsobeného pro vyhledávací stroj. Pro zpracování požadavku Lucene poskytuje několik způsobů a některé z nich budou popsány v následující sekci. Jak pro zpracování, tak i pro vyhledávání je potřeba konkrétního analyzátoru. Ideální a doporučovanou volbou je použít analyzátor se stejnými vlastnostmi pro obě fáze, to znamená pro proces indexování i vyhledávání. Použití jiného analyzátoru by mohlo vést ke snížení účinnosti vyhledávání až k nulové účinnosti. K hledání relevantních dokumentů v indexu slouží třída `IndexSearcher` z knihovny Lucene, která z důvodů stručnosti této kapitoly nebude dále popisována. Knihovna Lucene na dotaz vyhledávání (například zavoláním metody `search(Query query, Collector results)`) vrátí nejrelevantnější množinu výsledků. Velikost této množiny se nastavuje jako vlastnost objektu `Collector`. Vyhodnocení jaké dokumenty jsou nejrelevantnější k zadanému dotazu záleží na vytvořeném indexu a hodnota konkrétního „skóre“ dokumentu se také odvíjí od použitého analyzátoru.

3.4.5 Sestavení dotazu k vyhledávání

Lucene poskytuje několik prostředků k sestavení dotazu. Dotaz se skládá z operátorů a termů, kterými je určena jeho syntaxe. Lucene umožňuje pro konstrukci dotazu využít booleovské operátory, fuzzy logiku, proximitní hledání, metod k hledání v definovaném rozsahu a využití zástupných znaků. Níže jsou uvedeny významy jednotlivých konstrukcí, přičemž zdrojem některých příkladů je [Dot(2013)].

- **Booleovské operátory** - V tomto případě se jedná o obecně známé booleovské operátory AND, OR a NOT. Také se dají sdružovat jednotlivé termy pomocí závorek. Například *(auto OR dum) AND vlastník*.
- **Fuzzy logika** - Zástupný znak pro fuzzy logiku v Lucene je „∽“. Příkladem použití fuzzy logiky je výraz *zet'∽0.5*, který vyhledá a ohodnotí podobná slova na základě lexikální analýzy.
- **Proximitní hledání** - Jako fuzzy logika i proximitní hledání používá označení „∽“, ale výraz před vlnovkou musí být uzavřen v uvozovkách. Například dotaz *"jakarta apache"∽10* vyhledává slova „jakarta“ a „apache“ ve vzdálenosti maximálně deseti slov.
- **Definování rozsahu** - Definování rozsahu hodnot se zapisuje do hranatých závorek. Příklad hledání hodnot nacházejících se v rozmezí od 10 do 20 je výraz *pole:[10 TO 20]*. Obdobně lze nastavit rozmezí pro textové řetězce.
- **Zástupné znaky** - V dotazu lze použít také zástupné symboly (wildcard), kterými jsou „?“ a „*“. Otazník odpovídá jednomu libovolnému znaku, kdežto hvězdička nahradí sekvenci znaků. Například dotaz *te?t* nalezne shodu u slov *text* a *test*.
- **Boosting výrazu** - Nastavením boost faktoru na hodnotu vyšší než jedna, například pro slovo auto (*auto^3*) se explicitně vynutí, aby toto slovo mělo větší míru relevantnosti při prohledávání v indexu.

Analýza dotazu

Před popisem analyzování (parsování) dotazu je nutné objasnit několik faktů. Lucene umožňuje použít také jiné operátory, než jsou výše popsány, a kterými lze snadněji sestavit výsledný dotaz předaný odpovídající metodě. Jedná

se o operátory plus a mínus. Operátor plus před dotazem (`+apache`) znamená, že slovo „apache“ se musí vyskytovat v dokumentu. Protikladem k operátoru plus je mínus, tedy zaměněním plus za mínus by mělo za následek vyloučení dokumentů obsahujících slovo „apache“.

Jelikož se k ukládání dat používají pole, je možné zadávat dotazy zaměřené na vyhledávání v konkrétním poli, například dotaz *title:”Lucene cookbook”* zaručí vyhledávání publikací s názvem „Lucene cookbook“ (uzavření textu do uvozovek zajistí vyhledávání přesné fráze v indexu) v poli nazvaném „title“.

Kromě výše uvedených způsobů sestavení dotazů viz sekce 3.4.5, Lucene nabízí analyzování neboli parsování dotazu zavoláním metody `parse(String)` nad vytvořeným objektem třídy `QueryParser`. Syntaxe řetězce dotazu je následující. Dotaz je série klauzulí. Klauzule může mít prefix plus, mínus nebo termín následovaný dvojtečkou viz výše vyhledávání v konkrétním poli. Klauzulí může být bud': termín, na základě kterého se označí všechny dokumenty, které obsahují tento termín; nebo vnořený dotaz uzavřený v závorkách, přičemž může mít s +/- prefix. Syntaxi dotazu v Backusova-Naurova formě lze zapsat[12]:

```
Query ::= ( Clause )*
Clause ::= ["+", "-"] [<TERM> ":"] ( <TERM> | "("Query ")")
```

3.4.6 Analyzátor

Abstraktní třída `Analyzer` specifikuje metody používané k analyzování dat, jinými slovy definuje způsob získání jednotlivých slov neboli termů určených k indexování. Lucene obsahuje několik již implementovaných analyzátorů, jejichž výčet je níže.

- Analyzátor `WhitespaceAnalyzer` je jeden z několika analyzátorů obsažených v Lucene. Tento analyzátor rozdělí text do jednotlivých tokenů podle bílých znaků a nevykonává žádnou další úpravu textu.
- `SimpleAnalyzer` analyzátor rozdělí text na tokeny podle znaků, které nejsou abecední (regulární výrazem je vyjádřeno jako `[^A-Za-z]`), a potom převede všechna velká písmena na malé.

[12]Převzato z [Bnf(2012)].

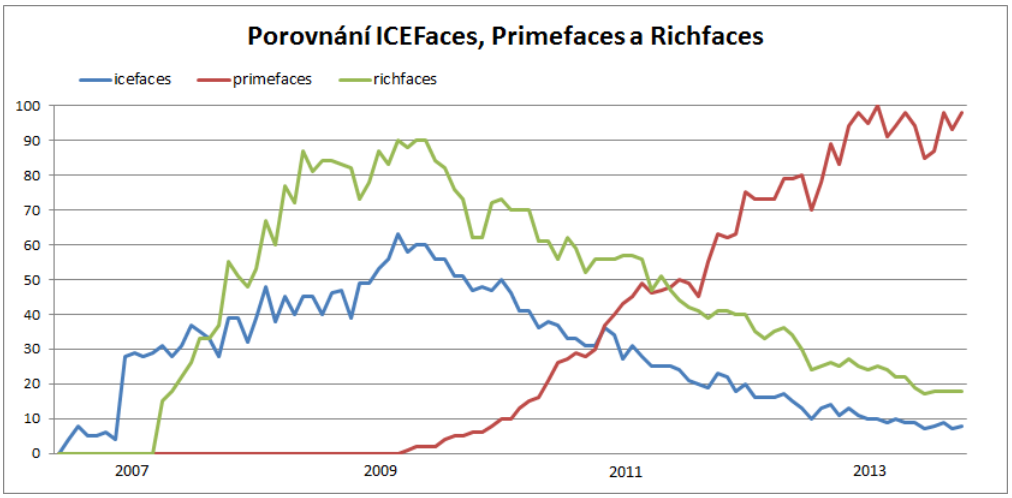
- Analyzátor `StandardAnalyzer` rozděluje text na jednotlivá slova s tím, že všechna písmena ve slově budou převedena na malé a odstraní stop slova. Tento analyzátor také rozeznává čísla s desetinou čárkou, IP adresy, emailové adresy, akronymy, názvy společností a další.
- `StopAnalyzer` je podobný jako `SimpleAnalyzer` kromě toho, že odstraní stop slova.

Kromě využití hotových analyzátorů lze také implementovat vlastní a to s pomocí tzv. filtrů. Pro účely této práce byl vytvořen `StemmAnalyzer`, jenž disponuje stejnými vlastnostmi jako standardní analyzátor s přidanou vlastností poskytnutou filtrem `PorterStemFilter`. Filter `PorterStemFilter` propůjčuje analyzátoru vlastnost **stemmatizace**.

3.5 Primefaces

Primefaces je frontendový framework pro javovské webové aplikace a byl zvolen pro vývoj cílové aplikace této práce. Primefaces je knihovna komponent pro JavaServer Faces (JSF) pod licencí Apache 2.0. Hlavními konkurenty v této oblasti jsou **RichFaces** a **IceFaces**. Na následujícím grafu[13] je znázorněno jak si Primefaces vede v posledních několika letech v „oblíbenosti“ ve srovnání s ostatními frameworky. Hodnoty na svislé ose reprezentují oblíbenost vyhledávaného dotazu. Konkrétní vysvětlení významu hodnot v grafu je na stránkách Google Trends. V následujícím textu budou diskutovány některé výhody a nevýhody zmíněných frameworků, přičemž některé pojmy nejsou přeloženy do českého jazyka. Podrobné srovnání Primefaces s jinými alternativami je v [Marchioni(2012)].

[13]Graf na obr. 3.2 je převzat z Google Trends - `http://www.google.cz/trends/`.



Obrázek 3.2: Srovnání frontendových frameworků podle Google Trends.

Primefaces obsahuje okolo 117 základních komponent včetně několika jejich variant, které zahrnují i další „nadstandardní“ komponenty. Mezi tyto nadstandardní komponenty se řadí: html editor; grafy; export dat do excelu a další. Primefaces komponenty v pozadí používají knihovnu jQuery a technologii Ajax a s jejich pomocí vytváří cílový vzhled. Tento framework také poskytuje více grafických motivů než Icefaces a Richfaces.

Primefaces poskytuje dokument s celou řadou návodů a příkladů použití knihovny. Rozběhnutí začátku vývoje aplikace s touto knihovnou je velmi jednoduché, což se se uvádí jako jedno z pozitiv u Primefaces, neboť je potřeba jen jeden *jar* soubor bez dalších závislostí. Ale jsou nutné další závislosti k nahrávání souborů, pro manipulaci s excelem nebo s dokumenty ve formátu `pdf`.

Alternativa Richfaces

Richfaces obsahuje 39 základních komponent a „variant“ (11 základních komponent, 6 panelů, 9 tables-grid,1 tree, 4 toolbar a 8 menu). RichFaces nabízí nástroj Component Development Kit (CDK), který usnadňuje práci s těmito komponenty. Ovšem slabou stránkou Richfaces je relativně málo ukázkových příkladů použití komponent.

Součástí dokumentace není návod s popisem jak začít využívat tuto knihovnu (tzv. kickstart tutorial). Začátek vývoje pomocí Richfaces není příliš složitý. K fungování vyžaduje základní knihovnu společně s UI knihovnou a tři nutné závislosti na externích zdrojích. Další závislosti jsou volitelné.

Alternativa Icefaces

Icefaces obsahuje okolo 70 základních komponent. Výhodou tohoto fameworku je možnost použití dalšího modulu ACE Components (ICEfaces Advanced Components), což je další generace Icefaces komponent. V současné době se najde přes 40 komponent zahrnujících `ace:dataTable`. ACE komponenty využívají kombinaci vykreslovacích technik na straně serveru (server-side) a webového klienta (client-based) k dosažení cílového vzhledu grafického rozhraní a tím se zredukují nároky na výkon zařízení.

Pro Icefaces je vytvořen dokument s návody a příklady. K vývoji vyžaduje několik málo knihoven a závislostí (většinou běžné Apache knihovny).

Zhodnocení a výběr

Zdůvodnění výběru Primefaces bylo podpořeno množstvím dostupných komponent, na základě zmiňovaných vlastností a rozmanitostí grafických motivů, které umožňují relativně rychle vytvářet pěkné a dynamické grafické rozhraní.

3.6 Maven

Apache Maven[14] je nástroj určený ke správě, řízení a automatizaci sestavování aplikace. Maven podporuje hlavně programovací jazyk Java a pokrývá následující oblasti.

- Usnadnění procesu sestavování aplikace,
- jednotný systém sestavování aplikace,

[14]Podrobná dokumentace [Mav(2014)].

- poskytování kvalitních informací o projektu,
- poskytování direktiv pro „best practices“,
- poskytnutí transparentního přidávání nových funkcí.

Struktura aplikace a popis projektu je řízen pomocí Project Object Model, který popisuje softwarový projekt z pohledu zdrojového kódu, závislostí na externích knihovnách a také popisuje proces sestavování aplikace společně s dalšími podpůrnými funkcemi. Project Object Model je definován XML strukturou, která určuje jednotlivé části projektu a závislosti na externích knihovnách atd. Také je možné vytvářet konstanty pro potřeby jednotlivých pluginů. Tento XML dokument je uložen pod názvem *pom.xml* a pro každý dílčí modul/projekt existuje jiný pom.xml soubor, který přebírá vlastnosti od nadřazeného souboru.

Alternativou k Maven je nástroj **Apache Ant**, kterým lze automatizovat činnosti: kompilace, testování, vytvoření balíku pro distribuci a další. Princip tohoto nástroje je stejný s unixovým nástrojem **Make**. Skripty pro Ant jsou psány v jazyce XML a tento nástroj je napsaný v jazyce Java, což znamená jeho platformní nezávislost. Oproti Maven postrádá několik funkcí navíc. Další informace o nástroji jsou k nalezení na [Ant(2014)].

Zhodnocení nástrojů je následující. Oba nástroje jsou podporovány řadou vývojových nástrojů. Nicméně z důvodů vyplývajících z výše uvedeného popisu byl zvolen nástroj Maven hlavně z důvodu jednoduché správě závislostí na externích zdrojích a pro co nejsnazší migraci projektu na jiné vývojové prostředí.

3.7 Tomcat

Apache Tomcat je aplikační server vyvíjený pod licencí open source. Tento server je založen na programovém jazyce Java, javových servletech, Java Server Pages (JSP) a Enterprise JavaBeans (EJB). Alternativy k tomuto serveru jsou například GlassFish, Jetty a další. Tomcat neposkytuje velkou robustnost a propracovanou bezpečnostní stránku jako jiné komerční produkty, ale na druhou stranu je jednoduchý, transparentní, má menší nároky na výpočetní výkon a umožňuje rychlejší vývoj. Volba Tomcatu jako aplikačního serveru je podmíněná jeho rozumnou kvalitou, velkou uživatelskou

základnou a relativně rychlou možností vývoje, i když v některých ohledech zaostává za konkurencí. Následuje porovnání jen některých vlastností zmíněných serverů. Podrobné srovnání lze nalézt v [Tom(2014)].

GlassFish a Tomcat

GlassFish má téměř kompletní podporu pro Java 5 a 6 EE, kdežto Tomcat podporuje jen servlety a JSP standardy. Všechny webové služby určené pro práci na Tomcat lze spustit i na GlassFish. GlassFish poskytuje více nástrojů k administraci a je vyvíjen společností Oracle, zatímco Tomcat je projekt Apache Foundation. Po výkonnostní stránce jsou si podobné. Další rozdíly lze nalézt v [Gla(2013)].

Jetty a Tomcat

Tomcat mívá o něco málo lepší výkon, je-li navázáno několik velmi aktivních připojení. Oproti tomu Jetty má lepší škálovatelnost je-li navázán větší počet připojení s delší dobou nečinnosti. Jetty má také malé paměťové nároky a servlety spotřebovávají méně vyrovnávací paměti procesoru. Více informací se nachází v [Jet(2013)].

3.8 XML parser

XML parsery usnadňují čtení a kontrolu syntaxe XML dokumentů. Pro čtení XML databáze (viz kapitola 2) vzhledem k její velikosti bylo zvoleno rozhraní SAX (Simple API for XML). Pro další práci s XML dokumenty kromě databáze bylo zvoleno to samé rozhraní.

Podle [Sax(2014)] výhoda SAX parseru je v událostmi řízeném přístupu, což spotřebovává málo paměti a zpracování je rychlé. Události jsou vyvolávány v pořadí tak, jak jsou čteny jednotlivé elementy v dokumentu.

Alternativou k rozhraní SAX je použití rozhraní Document Object Model (DOM), které vyžaduje nejprve načtení celého dokumentu, než se s daty může pracovat. DOM reprezentuje dokument jako stromovou hierarchickou strukturu, kde každý element je zastoupen jedním uzlem stromu. Toto rozhraní

umožňuje vytvořenou strukturu procházet, modifikovat uzly, mazat a přidávat uzly. Oproti rozhraní SAX spotřebovává více paměti a je pomalejší ve zpracování dokumentu. Více informací se nachází v [Dom(2011)].

Digester

Ke zjednodušení parsování XML databáze byla zvolena knihovna *digester* a vzhledem k jednoduché struktuře dat není náročné tuto knihovnu použít. Apache Commons Digester je podle [Dig(2014)] Java knihovna, která převádí XML data do javovských objektů. Jejím použitím se programátor vyhne přímému použití SAX API. Digester uvádí tři důležité pojmy: processing rules (pravidla zpracování), object stack (objektový zásobník) a element matching patterns (element odpovídající vzorům). Pod pojmem „element matching patterns“ se skrývá asociace XML elementů s procesními pravidly, viz následující příklad.

```
1  <items>        'items'
2    <item>       'items/item'
3      <name/>    'items/item/name'
4      <age/>     'items/item/age'
5    </item>
6    <item>       'items/item'
7      <name/>    'items/item/name'
8      <age/>     'items/item/age'
9    </item>
10 </items>
```

Z příkladu uvedeného výše lze vypozorovat, že prvek `'items/item'` asociovaný s daným pravidlem se spustí dvakrát.

Processing rules definují co se má provést, když se najde shoda podle vzoru. Digester zahrnuje několik předefinovaných processing rules, ale je možné vytvořit i vlastní pravidla.

Object stack zpřístupňuje objekty pro manipulaci prostřednictvím „processing rules“. Objekt může být přidán nebo odstraněn ze zásobníku manuálně anebo prostřednictvím „processing rules“.

4 Způsob indexace a sestavení dotazu

V této kapitole bude vysvětleno, jakým způsobem jsou použité prostředky popsané v kapitole 3 zakomponovány ve výsledné aplikaci. Pro indexaci dat a následné vyhledávání kurzů je použit analyzátor `StemmAnalyzer` popsaný v sekci 3.4.6.

První část kapitoly bude zaměřena na popis indexování databáze pomocí prostředků, které nabízí Apache Lucene. V druhé části je popsáno jakým způsobem jsou prostředky lemmatizátor, tezaurus a ACM CCS zakomponovány do výsledného algoritmu pro sestavování dotazu použitém v aplikaci.

4.1 Způsob indexování databáze

Indexování výsledné databáze popsané v kapitole 2 je realizováno pomocí knihovny `IndexCoursesXML`, která využívá knihovnu Lucene. Implementace navrženého řešení indexování je uvedena v sekci 5, a proto zde nebudou detaily týkající se programového kódu kromě několika příkladů.

Indexování dat je značně usnadněno knihovnou Lucene. Lucene poskytuje pro nastavení boost faktoru metodu `setBoost(float)` ze třídy `org.apache.lucene.document.Field`. Textový obsah je možné ukládat do vzniklé struktury indexu jako objekt typu `TextField` odvozený od třídy `Field` společně s nastavením způsobu uložení informací. Konstruktoru `TextField(String, String, Store)` třídy `TextField` lze nastavit způsob uložení informací pomocí parametru, kterým je výčtový typ (`Store`) třídy `Field`. Výčtový typ `Store` nabízí volby `YES` a `NO`.

Zvolení jakým způsobem se mají data indexovat je silně ovlivněno strukturou a významem dat, proto se důležité úseky dat zvýrazňují přenastavením boost faktoru a ostatním se ponechá výchozí hodnota. Databáze dat je podrobně popsána v kapitole 2 a veškerý následující text v této kapitole se bude vztahovat ke zmíněné databázi. Ve struktuře XML se nacházejí, kromě jiných, elementy *overivew* a *title*, které nejvíce vypovídají o obsahu daného kurzu. Následuje popis výše vypsaných elementů a odůvodnění odlišného přístupu k indexaci od ostatních elementů.

Indexování elementu *overview*

Tento element, jak je popsáno v kapitole 2, lze obsahem přirovnat ke sylabu předmětu/kurzu na univerzitě. Nejčastěji tento element obsahuje text o délce patnácti až pětatřiceti slovech viz histogram 2.1. Vzhledem k charakteru elementu je považován za jeden nejdůležitějších, protože nejlépe vystihuje obsah kurzu. Na základě zmíněných skutečností se tomuto elementu prostřednictvím nabízených prostředků Lucene nastaví jiné ohodnocení. Ohodnocení v terminologii Lucene se označuje jako *boosting* a pro element overview je nastaven na hodnotu „3.0f“.

Indexování elementu *title*

Tento element obsahuje název kurzu. Název kurzu má také určitou vypovídající hodnotu o obsahu kurzu, avšak ve většině porovnání s elementem overview není tak významná. I u tohoto elementu stejně jako u overview je nastaven boost faktor na vyšší než běžnou hodnotu tzn. je nastaven na hodnotu „2.0f“.

Kromě zmíněných elementů overview a title mají všechny elementy nastaven boost faktor na výchozí hodnotu určenou použitou knihovnou Lucene. Obsah všech elementů je ukládán z důvodu potřeby znovu získání původního obsahu k zobrazení výsledků, což samotná indexace nezajišťuje.

4.2 Strategie sestavení dotazu

Tato sekce je věnována popisu sestavení výsledného dotazu, který se nakonec použije jako vstupní dotaz pro vyhledávání v elementu *overview*. Rozšiřování se provádí ve dvou krocích, přičemž druhý krok je závislý na prvním. Prvním krokem je rozšíření dotazu pomocí ACM CCS a ve druhém kroku se použije tezaurus. Zdůvodnění rozšiřování (sestavení) dotazu je v kapitole 3.1. Vyhledávání je realizováno pomocí prostředků knihovny Lucene.

Metoda jakou se dotaz rozšiřuje se odvíjí od zvolených možností k zadání dotazu prostřednictvím grafického rozhraní, které proto bude v následujících větách naznačeno. Aplikace disponuje grafickým rozhraním, které nabízí vyhledávání v elementu *overview* třemi způsoby. Je tedy možné zadat slova

nebo fráze (fráze jsou zadány v uvozovkách), které se v hledaném elementu:

- musejí vyskytovat, tzn. nastaví se v Lucene vlastnost `MUST`,
- nemusejí vyskytovat, tzn. nastaví se v Lucene vlastnost `SHOULD`,
- nesmějí vyskytovat, tzn. nastaví se v Lucene vlastnost `NOT`.

Samotné rozšíření se provádí nad jednotlivými slovy a jen pokud jsou zadané jako povinné výrazy. Zadané fráze se ponechají v původním tvaru. Následuje postup rozšiřování dotazu pomocí ACM CCS a WordNet tezauru. Prvním krokem je získání seznamu jednotlivých slov z dotazu zadaném tazatelem. Potom se na všechny slova ze seznamu aplikují následující kroky. Slova v seznamu se lemmatizují[1] za účelem získání jednotného tvaru k maximalizování efektivnosti rozšiřování dotazu. Ještě před dalším postupem jsou odstraněny z tohoto seznamu slova, která jsou součástí *stoplistu* viz začátek kapitoly 3. Získaná slova tímto způsobem jsou označena jako klíčové slova v dotazu. Další kroky se liší v závislosti na použití ACM CCS nebo tezauru.

V dalších krocích, které jsou jedinečné pro rozšiřování dotazu pomocí ACM CCS se nejprve přistoupí k porovnávání hledaných slov (klíčových slov) s pojmy z daného XML dokumentu ACM CCS. Z důvodů maximalizování shody porovnávání pojmů z ACM CCS s daty v prohledávaném indexu se na pojmy aplikuje lemmatizace a poté jsou odstraněna stop slova. Po nalezení shody jsou do rozšíření dotazu zahrnuty hyponyma neboli bezprostředně podřazené pojmy a souřadné pojmy jsou vynechány. Důvodem k omezení rozšíření dotazu jen o hyponyma je, aby nebyla množina získaných pojmů příliš velká viz odstavec „Kritéria výběru“ v sekci 3.3. Po první shodě a získání všech hyponym je algoritmus ukončen. Pro úplnost je níže uveden v krocích naznačený postup s příkladem na konci.

[1] Použitý lemmatizátor je popsán v sekci 3.1.

4.2.1 Postup rozšíření s ACM klasifikačního systému

Zadaný dotaz je „protocols and networks "computer science"“. Následuje postup rozšíření dotazu popsaný v osmi bodech.

1. Odstranění nežádoucích znaků „: ; , . ! ? < >“.
2. Z dotazu se vytvoří seznam z jednotlivých slov získaných z původního dotazu. Dotaz se rozdělí na slova a fráze podle regulárního výrazu `([^"]\S*|".+?")\s*`, který je předaný metodě `compile` třídy `java.util.regex.Pattern` v programovém jazyce Java[2]. Seznam tedy bude obsahovat pojmy *protocols*, *and* a *networks* ze zadaného dotazu.
3. Nyní se provede lemmatizace slov ze seznamu.
4. Odstraněná slova ze seznamu korespondující se slovy ze stoplistu.
5. Potom započne vyhledávání odpovídajících pojmů z ACM CCS ke kombinaci slov *protocol* a *network*. Při tomto kroku jsou pojmy z ACM CCS lemmatizovány a zbaveny tzv. stop slov[3].
6. Po shodě s pojmem „Network protocols“ z ACM CCS se vyberou hyponyma „Network protocol design“, „Protocol correctness“ atd. Vybraná hyponyma se ponechají v původním tvaru.
7. Vybrané pojmy rozšíří původní dotaz s tím, že nové pojmy v rozšířeném dotazu mají nastavenou vlastnost na SHOULD tzn. nemusejí se vyskytovat ve výsledku hledání.
8. Novým frázím se navíc nastaví proximita na hodnotu dva (`"example word"`$\backsim$`2`) a slovům se nastaví příznak fuzzy logiky s výchozí hodnotou[4] (`example`$\backsim$).

Příklad rozšíření dotazu

Před rozšířením by byl dotaz předaný metodě pro vyhledávání složen s následujících částí[5]:

[2]Verze programového jazyka Java je 1.7.

[3]Stop slova jsou slova ze seznamu stoplist.

[4]Výchozí hodnota fuzzy logiky se liší od použité verze knihovny Lucene a v dalších verzích není garantována stejná hodnota .

[5]Syntaxe dotazu je popsaná v kapitole 3.4.

- `+overview:protocols`,
- `+overview:networks`,
- `+overview:"computer science"`.

Po úpravě dotazu procesem rozšiřování se k původnímu dotazu přidají ještě nalezené pojmy z ACM CCS (níže uvedený výčet je omezen z důvodů přehlednosti):

- `overview:"Network protocol design"`⌣2,
- `overview:"Protocol correctness"`⌣2.

Po výše uvedeném postupu vysvětlujícím případné rozšíření dotazu pomocí klasifikačního systému následuje postup rozšíření pomocí WordNet tezauru. K použití tezauru dojde, pokud v předchozím postupu **nedošlo** k žádnému rozšíření dotazu.

4.2.2 Rozšíření dotazu pomocí nástroje WordNet

Vstupem tezauru je stejná množina jako pro rozšiřování pomocí ACM CCS. Rozšiřující pojmy se generují ke každému povinnému slovu. Bylo zvoleno, že rozšiřující pojmy budou získány ze synsetů typu podstatných jmen (`NOUN`). Z každého synsetu jsou všechny výrazy zařazeny do rozšiřování dotazu. Testováním, které spočívalo v získávání různých množin pojmů pomocí WordNet bylo zjištěno, že ostatní způsoby jsou nevyhovující viz sekce 3.2. Níže se nachází ukázka programového kódu, který má na starosti získání odpovídajících výrazů.

```
1  WordNetDatabase database = WordNetDatabase.getFileInstance();
2  Synset[] synsets = database.getSynsets(word, SynsetType.NOUN)
       ;
3
4  if (synsets.length > 0) {
5    String[] wordForms;
6    for (int i = 0; i < synsets.length; i++) {
7      wordForms = synsets[i].getWordForms();
8      addWordsToList(relativesWords, wordForms);
9    }
10 }
```

Výše uvedená ukázka vrátí pro slovo „network“ množinu pojmů (pojmy shodující se zadaným slovem se odstraní): *web*, *net*, *mesh*, *meshing*, *meshwork* a *electronic network*. Vrácená množina slov potom rozšíří dotaz podobným způsobem, jak je popsáno výše v části nazvané „Příklad rozšíření dotazu“ v této kapitole.

5 Implementace aplikace

Implementace výsledné aplikace je rozdělena do třech částí. Části popsané v sekci 5.2 a 5.3 jsou navrhnuté jako zásuvné moduly začleněné do výsledné webové aplikace. Takto je možné malou změnou v programovém kódu moduly pro indexaci a vyhledávání zaměnit. Zvolená verze knihovny Apache Lucene pro zásuvné moduly je 4.5.

5.1 Webová aplikace

Zvolenou platformou k vývoji webové aplikace je Java EE 1.7 a technologie zmíněné v kapitole 3. Konkrétní verze technologií, nástrojů a knihoven jsou: Tomcat 7.0, Primefaces 4.0, JSF 2.1 a další, které jsou uvedeny v souboru *pom.xml* ve zdrojových kódech aplikace.

Softwarovou architekturou aplikace je Model-view-controller (MVC)[1], podle které je aplikace rozdělena na tři úrovně:

- **Model** - reprezentace dat,
- **Controller** - reakce na události, řízení,
- **View** - zobrazení dat uživateli (prezentační vrstva).

Tato MVC architektura byla vzorem pro návrh aplikace, nicméně při implementaci nešlo zcela striktně dodržet hranice mezi vrstvami a dochází někdy k prolínání těchto hranic. Následující popis rozdělení jednotlivých částí je jednou variantou z několika možných.

5.1.1 Vrstva *model*

Všechna data o kurzech jsou v aplikaci udržována v instanci třídy `Database-Courses`, jež obsahuje tři seznamy s objekty třídy `Couse`[2]. Třída `Database-Courses` zapouzdřuje tři seznamy: seznam nalezených kurzů; seznam kurzu

[1] Informace o MVC architektuře jsou uvedeny v [Mvc(2014)]

[2] Třída `Course` je z knihovny `SearchCourses`

se zvýrazněnými klíčovými slovy (klíčová slova jsou ohraničena tagem `<span style="color: ...">`) v sylabu daného kurzu; seznam filtrovaných kurzů (vyžadováno Primefaces komponentou `p:dataTable`).

Data v této vrstvě jsou získávána z indexu vytvořeného z XML databáze kurzů. Přímou manipulaci s indexem, tzn. vytváření indexu a vyhledávání dat, obstarávají knihovny *IndexCoursesXML* a *SearchCourses*.

Dalšími strukturami (*java.util.List* a *java.util.Map*) udržujícími data pro prezentační vrstvu jsou: *axiliaryWords*, *selectedUniversities*, *universities*, *selectedYears*, *yearOffered*.

K uchování stromové struktury slouží struktura *TreeNode* pojmenovaná „root“, v níž jsou na první úrovni vnoření univerzity a na další úrovni katedry. Tato stromová struktura je přístupná přes `DepartmentTreeBean` tzv. *backing bean*.

5.1.2 Vrstva *view*

Prezentační vrstva je vytvořena technologií JSF a pomocí komponent Primefaces. Do této vrstvy se řadí webové stránky uvedené níže s krátkým popisem jejich účelu:

advancedsearch.xhtml	- formulář pro pokročilé hledání,
department.xhtml	- výběr kurzů podle katedry,
index.xhtml	- formulář pro základní hledání,
resultOfSettings.xhtml	- nastavení aplikace,
settings.xhtml	- výsledek úspěšnosti nastavení aplikace,
viewcoursepanel.xhtml	- zobrazení kurzů pomocí panelů (`p:panel`),
viewcoursetable.xhtml	- zobrazení kurzů pomocí tabulky,
viewdepartmentdialog.xhtml	- zobrazení kurzů pomocí tabulky s dialogy,
viewdepartmentrows.xhtml	- zobrazení kurzů pomocí tabulky s náhledem.

Podrobný popis webových stránek z hlediska vzhledu a funkčnosti je součástí uživatelské dokumentace, viz kapitola 7. CCS (Cascading Style Sheets) styl je pro všechny stránky jednotný a relativně jednoduchý. CSS styl je uložen v souboru *mstyle.css*. Textový obsah na všech stránkách je dostupný přes tzv. *backing bean* a je uložen v *properties* souborech s názvem, který je odvozen od názvu konkrétní stránky (Výjimkou je stránka *advancedsearch.xhtml*,

protože obsahuje část ze stránky *index.xhtml*). Například stránce *index.xhtml* odpovídá soubor s názvem *index_en.properties*. Rozdělení textového obsahu webových stránek umožňuje relativně jednoduchou záměnu lokalizace. V následujících řádcích je uveden popis stránek především z hlediska přiřazených *backing bean* z balíku `cz.zcu.kiv.dip.bean`.

Stránka *index.xhtml*

Na této stránce je podstatný formulář s možností zadání hledacích kritérií. Tato stránka je obsluhována `SearchingCoursesBean` a `UniAndYearBean`. Přes třídu `UniAndYearBean` jsou dostupné statistiky o databázi, což je seznam všech univerzit a let, ve kterých se kurzy vyučovaly z celé databáze. Třída `SearchingCoursesBean` má na starosti obsluhu formuláře nacházejícího se na této stránce. Po zpracování vstupních dat z formuláře přepošle dotaz na stránku *viewcoursepanel.xhtml*.

Stránka *advancedsearch.xhtml*

Rozšiřuje formulář ze stránky *index.xhtml* o podrobné možnosti zadání kritérií vyhledávaní. Tato stránka je obsluhována `AdvancedSearchingCoursesBean` a `UniAndYearBean`. Účel třídy `AdvancedSearchingCoursesBean` je stejný jako `SearchingCoursesBean`, ale obsahuje více polí pro nastavení dalších kritérií a má na starosti obsluhu formuláře nacházejícím se na této stránce. Po zpracování vstupních dat z formuláře přepošle dotaz na stránku *viewcoursepanel.xhtml*.

Stránka *department.xhtml*

Tato stránka zobrazuje data o dostupných katedrách ve formě stromové struktury. Tato struktura obsahuje katedry seřazené podle univerzit. Pro vybrané katedry se potom naleznou odpovídající kurzy a zobrazí se na stránce *viewdepartmentrows.xhtml*. Obsluhující třídou této stránky je `DepartmentTreeBean`.

Stránka *settings.xhtml*

Tato webová stránka slouží k nastavení cest k datům aplikace atd. Dále je možné zadat cestu ke zdrojové databázi kurzů, viz kapitola 2, společně s dalšími příslušnými parametry a provést indexaci XML databáze tzn. vytvořit nová data potřebná pro běh aplikace. Po nastavení nebo indexaci je zobrazena stránka *resultOfSettings.xhtml* s výsledkem akce. Třída `IndexFormBean` obsluhuje část formuláře určenou pro indexaci a třída `SettingsFormBean` obsluhuje druhou část formuláře pro účely nastavení parametrů aplikace. Zmíněné třídy slouží také pro *resultOfSettings.xhtml* k zobrazení výsledku provedené akce na stránce *settings.xhtml*.

Stránky *viewcoursepanel.xhtml* a *viewcoursetable.xhtml*

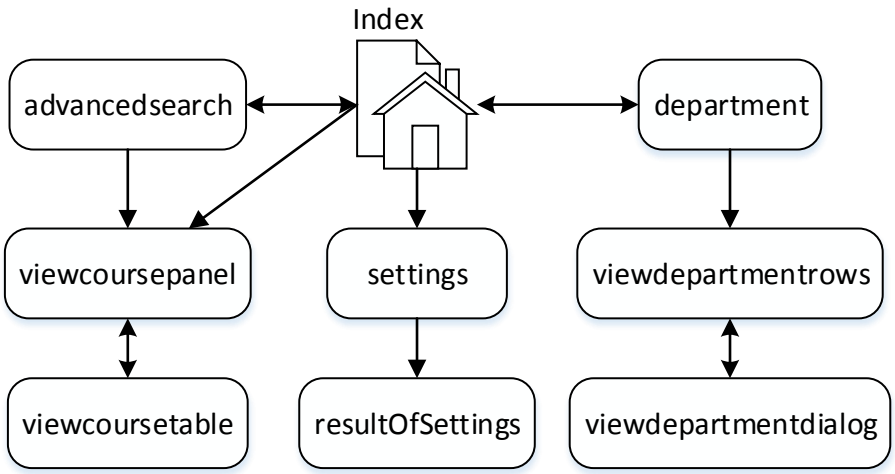
Tyto stránky obsluhuje třída `ResultSearchBean`, která poskytuje data o vyhledaných kurzech skrze objekt třídy `DatabaseCourses` a seznam pomocných pojmů pro vyhledávání. Třída `UniAndYearBean` zpřístupňuje pro tyto stránky seznam univerzit, ve kterých bylo vyhledáváno.

Stránky *viewdepartmentdialog.xhtml* a *viewdepartmentrows.xhtml*

Tyto stránky obsluhuje třída `ResultDepartmentBean` a stejně jako v předchozím případě zpřístupňuje data o kurzech prostřednictvím třídy `DatabaseCourses`. Třída `CourseDataModel` slouží k poskytnutí podrobných dat o kurzu prostřednictvím komponenty `p:dialog` z knihovny Primefaces.

Mapa webu

Na následujícím obr. 5.1 je zobrazena mapa stránek. Jedinou chybějící vazbou v obrázku je zpětný odkaz vedoucí z jakékoliv stránky na domovskou (*index.xhtml*).



Obrázek 5.1: Mapa webových stránek.

Vzhled v internetovém prohlížeči

Vzhledem k použitému frameworku Primefaces bylo dosaženo téměř stejného vzhledu webových stránek ve čtyřech internetových prohlížečích. Aplikace byla testována v prohlížeči Google Chrome (verze 34.0.1847.131 m), Internet Explorer (verze 11.0.9600.17105), Opera (verze 12.17) a Mozilla Firefox (verze 28.0).

Nejkorektnější vzhled tzn. všechny Primefaces komponenty se vykreslily podle vzoru, který byl v prohlížeči Mozilla Firefox. Rozdíly ve vzhledu v ostatních prohlížečích nejsou příliš velké na to, aby se ovlivnila funkčnost nebo přehlednost webu.

5.1.3 Vrstva *controller*

Na konci této sekce je umístěn doménový model z některých následujících tříd. Do této vrstvy se řadí tzv. *backing bean*, mezi které lze zařadit následující třídy.

SearchingCoursesBean

Tato třída obsluhuje formulář na stránce *index.xhtml* po zavolání metody `search()`, ve které se za pomoci metody `parseFieldsAndSetSearchingPhrases()` nastaví a zanalyzují vstupní pole pojmenovaná v aplikaci (na webové stránce) *Keywords and phrases* a *Any of these words*. Metoda `search()` dále zajišťuje získání dotazu a prostřednictvím jiné metody `setResultsProperties()` nastaví důležité data ve třídě `ResultSearchBean` a poté metodou `search(query)` na tu samou třídu deleguje požadavek na vyhledání kurzů. Pokud sestavení dotazu proběhlo v pořádku provede se přesměrování uživatele na *viewcoursepanel.xhtml*. Prostředky potřebné k analýze a vytvoření dotazu vyhovující syntaxi knihovně Lucene poskytuje třída `QueryBuilderUtils`.

AdvancedSearchingCoursesBean

Tato třída obsluhuje formulář na stránce *advancedsearch.xhtml* a je odvozená od třídy `SearchingCoursesBean`. Její úprava od rodičovské třídy spočívá mimo jiné v tom, že nejsou žádná povinná pole pro zadání kritérií k hledání a jedinou podmínkou vyhledávání je existence dotazu složený alespoň z jednoho kritéria. Dále rozšiřuje formulář o zadání dalších resp. všech možných kritérií k hledání kurzů. K sestavení výsledného dotazu využívá třídu `QueryBuilderUtils`.

ResultSearchBean

Třída `ResultSearchBean` je svázána se stránkami: *viewcoursepanel.xhtml* a *viewcoursetable.xhtml*. Používá přímo knihovnu `SearchCourses` k vyhledávání kurzů. Metoda `search()` využívá metodu k redukci kurzů ze třídy `UtilityReduceCourses`. Tato metoda dále zařizuje nastavení parametrů vyhledávání a poté i samotné vyhledání pomocí knihovny `SearchCourses`. Nakonec se analyzují nalezené kurzy a vyznačí se klíčová slova pomocí metody `setHighlightedCourses()` a takto upravené kurzy se uloží do instance třídy `DatabaseCourses`. Metoda ke zvýraznění klíčových slov využívá metod ze třídy `UtilityHighlightCourses`.

DepartmentTreeBean

Stránka *department.xhtml* je obsluhována touto třídou. V samotném konstruktoru se vytvoří datová struktura pro zobrazení dat, viz popis dané stránky v sekci 5.1.2. Metoda `viewCourses()` obsluhuje akci vyvolanou tlačítkem **View** z formuláře na této stránce. Tato metoda získá množinu vybraných kateder a vyvolá vyhledávání pomocí metody `search()` ze třídy `ResultDepartmentBean`.

ResultDepartmentBean

Mezi touto třídou a `DepartmentTreeBean` je obdobný vztah jako mezi třídami `SearchingCoursesBean` a `ResultSearchBean`. Metoda `search(List<UniAndDepartment> pars)` sestaví pro každou množinu kurzů rozdělenou podle kateder odpovídající dotaz vyhovující syntaxi knihovny Lucene a provede vyhledávání kurzů pomocí knihovny `SearchCourses`. Počet vyhledaných kurzů je omezen. Poté zredukuje[3] podobné kurzy a opakuje tento proces pro každou vybranou katedru. Tato třída je svázána se stránkami *viewdepartmentrows.xhtml* a *viewdepartmentdialog.xhtml*.

UniAndYearBean

Tato třída jak již bylo řečeno zpřístupňuje statistky o databázi kurzů, mezi které patří seznam univerzit a seznam let jednotlivých kurzů, ve kterých byly vyučovány nebo nabízeny. K načtení dat z XML souboru s těmito statistiky je použita třída `ReaderStatistics`.

SettingsFormBean

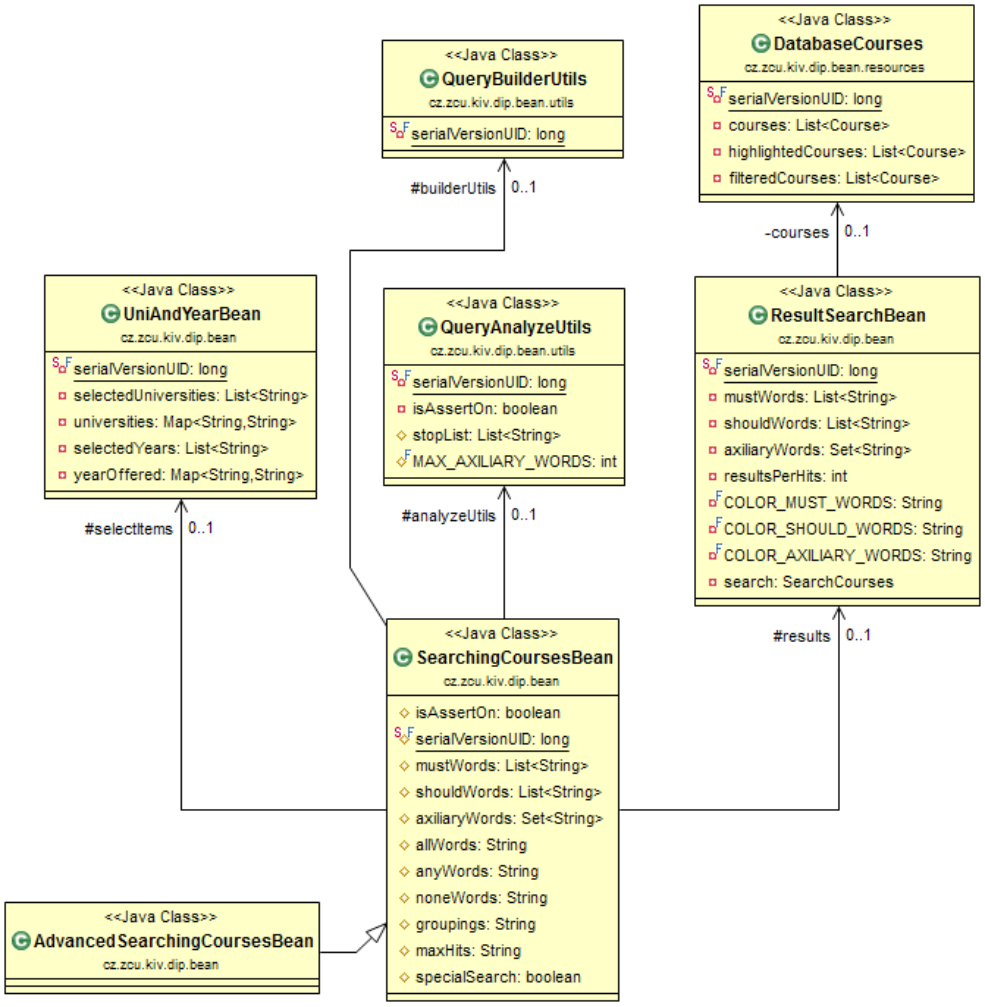
Tato třída společně s níže uvedenou třídou je spjata se stránkou *settings.xhtml* a využívá služeb třídy `UtilityValidSettings`. Jejím účelem je ověření zadaných cest a jejich následné nastavení.

[3]K procesu redukování podobných kurzů jsou použity prostředky třídy `UtilityReduceCourses`

IndexFormBean

Úkolem této třídy je ověření nastavených cest a vytvoření indexovaného dokumentu, ve kterém aplikace může vyhledávat. K indexování využívá knihovnu `IndexCoursesXML`.

Následuje doménový model na obr. 5.2 popisující vztahy okolo třídy `SearchingCoursesBean`. Některé třídy v doménovém modelu jsou popsané v sekci 5.1.4.



Obrázek 5.2: Doménový model vztahů okolo třídy `SearchingCoursesBean`.

5.1.4 Pomocné třídy

V této sekci budou popsány pomocné třídy využívané třídami popsanými v sekci 5.1.3. Třídy jsou rozdělené podle balíků. Každá z uvedených tříd má prefix *cz.zcu.kiv.dip*. V sekci 5.1.5 se nachází doménový model znázorňující provázanost různých tříd z hlediska probíhající komunikace mezi jednotlivými komponenty od zadání požadavku až po zobrazení výsledku. Tento model dovysvětluje a podává celkový obraz o provázanosti následujících popisovaných tříd a balíků.

Balík *bean.resources*

V tomto balíku jsou pouze třídy zapouzdřující data získaná z formulářů umístěné na stránkách *index.xhtml* a *advancedsearch.xhml* a data o vyhledaných kurzech. Třída `DatabaseCourses` obsahuje struktury k uložení vyhledaných a zvýrazněných kurzů, společně se seznamem vyfiltrovaných kurzů. Vyfiltrované kurzy jsou využívány komponentou `p:dataTable` Primefaces.

Balík *bean.utils*

V tomto balíku jsou třídy implementující strategii rozšiřování dotazu popisovanou v sekci 4.2. Třída `QueryAnalyzeUtils` poskytuje metody ke zpracování zadaného dotazu uživatelem, což zahrnuje extrakci hledaných slov a frází. Veřejnými metody třídy jsou:

- Metoda `getRequiredWordsAndSetKeyWords(...)` analyzuje předaný řetězec a vrátí seznam frází a lemmatizovaných slov získaných ze řetězce. Při analýze řetězce zároveň vytvoří seznam klíčových slov.
- Metoda `getAxiliaryWords(List<String> keyWords)` vyhledává shodné pojmy z množiny *keyWords* s pojmy z ACM CCS a v případě žádné shody přejde k použití tezauru WordNet. Jinak řečeno na základě slov keyWords rozšiřuje dotaz o pomocné výrazy.
- Metoda `getShouldWords(String anyWords)` provede stejnou analýzu jako předchozí metoda s tím rozdílem, že nevytváří seznam klíčových slov a **ne**aplikuje proces lemmatizování slov.

Součástí tohoto balíku je i třída `QueryBuilderUtils`. Metoda `getQuery(...)` této třídy sestavuje výsledný dotaz se syntaxí vyhovující knihovně Lucene. Tato metoda je přetížená. V jednom případě sestavuje dotaz na základě dat získaná z formuláře ze stránky *index.xhtml* a ve druhém případě ze stránky *advancedsearch.xhtml*. Ve druhém případě je dotaz rozšířen o všechna kritéria hledání (tzn. dotaz může obsahovat kritéria odpovídající každému elementu, který se může vyskytovat v kurzu).

Metoda `splitQueryIntoPhrases(...)` z pomocné třídy `QueryUtility` z téhož balíku rozdělí text předaný parametrem na jednotlivá slova a fráze.

Balík *properties*

Obsahuje dvě třídy `GlobalProperties` a `MessageProperties`, které slouží k přístupu textu různých zpráv a nastaveným cestám prostřednictvím stránky *settings.xhtml*. Odpovídající *properties* soubory jsou: *config.properties* a *message_en.properties*.

Balík *utilities*

Tento balík obsahuje pomocné třídy:

- Třída `Utility` obsahuje metody k odstranění nežádoucích slov, stop slov a číselných údajů z textu.
- Třída `UtilityHighlightCourses` disponuje logikou provádějící zvýrazňování hledaných slov v textu. Hledané fráze a slova jsou v textu zvýrazněny jen pokud se přesně shodují s úryvkem v textu anebo je podmnožinou ve smyslu řetězce, viz dále. Příklad zvýraznění: pomocná fráze je „networks protocol“ a úryvek textu je „using network protocols“, potom zvýrazněná část textu zahrnuje také písmeno „s“ na konci slova „protocols“ - using **network protocols**. Algoritmus pro zvýrazňování by se mohl zdokonalit, ale vzhledem k častému používání algoritmu byla upřednostněna jednoduchost a tím i rychlost.
- Třída `UtilityReduceCourses` slouží k redukování podobných kurzů z množiny. Kurzy jsou považovány za podobné, pokud se shoduje obsah elementů *overview*, *title*, *department* a *institution*. Redukováním

resp. sloučením se rozumí vytvoření nového objektu kurzu s daty obsaženými v obou kurzech. Například pro element *yearoffered* s obsahem „2011“ v jednom případě a „2012“ ve druhém případě se sloučí, přičemž výsledný obsah bude „2011, 2012“.

- Třída `UtilityValidPaths` obsahuje metody pro ověření existence složky nebo souboru.

Balík *statistics*

V této třídě `ReaderStatistics` je implementován proces analyzování XML souboru se statistiky o databázi kurzů pomocí knihovny `Digester` a data jsou shromažďovány v instanci třídy `StatisticsCollection`.

Balík *load*

Obsahuje třídy určené k načtení seznamu stop slov ze zdrojového souboru a třídy k získání pojmů z ACM CCS shodné s vyhledávanými klíčovými slovy. ACM CCS je uložen ve formě XML dokumentu a hledání shodných pojmů se realizuje při procesu analyzování klasifikačního systému s pomocí Java nástroje SAX parser.

Balík *resourses*

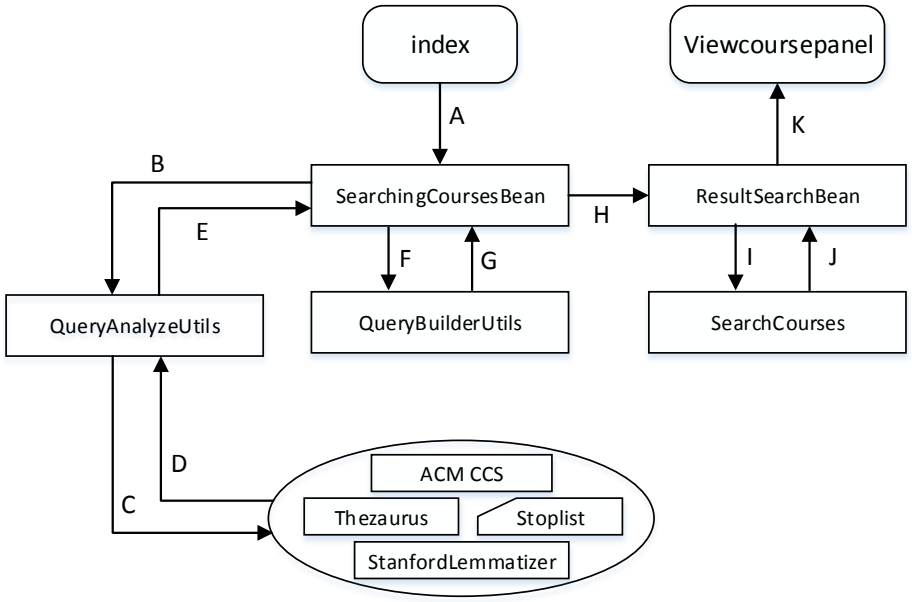
Tento balík obsahuje třídy, které zapouzdřují různá data a objekty těchto tříd slouží především jako pomocné objekty při výměně informací mezi třídami.

Balík *lemmatizer* a *thesaurus*

Balík *lemmatizer* obsahuje třídu `StanfordLemmatizer` sloužící k nastavení lemmatizátoru a disponuje metodou provádějící lemmatizaci textu. V baíku *thesaurus* se nachází třída `Thesaurus`, která obsahuje jednu veřejnou metodu `getSynonyms(String word)` jejíž návratová hodnota je seznam nalezených pojmů daných tezaurem.

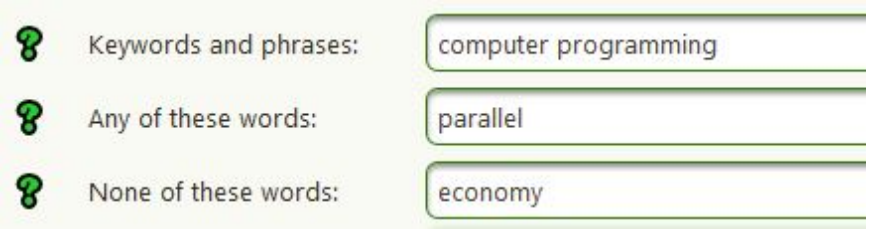
5.1.5 Cyklus obsluhy požadavku

Na následujícím obr. č. 5.3 je uveden diagram znázorňující provázanost různých tříd z hlediska probíhající komunikace mezi jednotlivými komponenty při obsluze požadavku na vyhledávání až po zobrazení výsledku. V diagramu jsou záměrně vynechány některé méně důležité třídy. Diagram odpovídá cyklu započatému zadáním požadavku k vyhledání kurzů přes základní formulář, který je dostupný na domovské stránce. Pod diagramem následuje jeho popis.



Obrázek 5.3: Diagram cyklu obsluhy požadavku vyhledávání.

V rámci jednoho přechodu může být provedeno hned několik procesů. V případě komunikace mezi bezprostředně navazujícími komponenty může dojít k opakované výměně dat, například přechody **B** a **E** v diagramu mohou být několikrát po sobě provedeny. Označení přechodů neudává/nezaručuje jejich pořadí. V diagramu třída s názvem `Thezaurus` zapouzdřuje nástroj WordNet. Tazatelem zadaná vstupní data jsou na následujícím obrázku:



Obrázek 5.4: Ukázka zadání některých vyhledávajících kritérií.

Následuje popis postupu cyklu obsluhy požadavku v osmi bodech.

1. V Rámci prvního **A** přechodu se předají data, tedy kritéria hledání zadaná tazatelem třídě `SearchingCoursesBean`. Data jsou získána prostřednictvím formuláře dostupným na stránce *index.xhtml*.

2. Během druhého přechodu **B** se předá první část dotazu *computer programming* zadaná tazatelem ke zpracování třídě `QueryAnalyzeUtils`. Část dotazu z pole *Keywords and phrases:* je vždy povinná. Jsou to tedy pojmy, jejichž výskyt je povinný v obsahu sylabu kurzu.

3. Třída `QueryAnalyzeUtils` komunikuje s pomocnými třídami prostřednictvím přechodů **C** a **D**. Výsledkem zpracování v tomto kroku je seznam povinných slov z původního dotazu. Tento seznam je zpět předán třídě `SearchingCoursesBean` přechodem **E**.

4. V dalším kroku se zpracuje část dotazu *parallel* stejným způsobem jako v přechozích krocích 2 a 3.

5. V tomto kroku se najdou tzv. pomocná slova nebo fráze (dále jen fráze) k povinným slovům. Seznam povinných slov je předán přechodem **B** třídě `QueryAnalyzeUtils`, která komunikuje přes přechody **C** a **D** s pomocnými třídami. Získané pomocné fráze se předají třídě `SearchingCoursesBean` přechodem **E**.

6. Ze získaných seznamů z předchozího zpracování a zbývajících zadaných kritérií se za pomoci třídy `QueryBuilderUtils` vytvoří výsledný dotaz se syntaxí vyhovující knihovně Lucene. Přechodem F se přesunou data do třídy `QueryBuilderUtils`, ze kterých se vytvoří výsledný dotaz. Následně se dotaz přepošle přechody **G** a **H** do třídy `ResultSearchBean`.

7. Ve třídě `ResultSearchBean` se shromáždí výsledky hledání získané prostřednictvím knihovny `SearchCourses`. Související přechody jsou **I** a **J**.

8. Shromážděné výsledky hledání jsou potom poskytnuty webové stránce *viewcoursepanel.xhtml* (přechod **K**).

5.2 Knihovna `SearchCourses`

Tato knihovna byla vytvořena na platformě Java SE. Je určena pro vyhledávání v indexu pomocí knihovny Lucene. Na konci této sekce je umístěn obr. 5.5 zobrazující doménový model[4] v němž jsou vynechány privátní atributy a ve třídě `Course` všechny getry a setry z důvodu jejich velkého počtu. Všechny třídy a rozhraní mají základní prefix:

```
cz.zcu.kiv.dip.lucene.search
```

a proto budou popisovány bez tohoto prefixu.

Rozhraní `Search`

Jedna z metod definovaná v tomto rozhraní je `public void setResultsPerHist(int ResultsPerHits)` skrze kterou se nastavuje maximální možná velikost nalezené množiny pro daný dotaz. Parametr nastavený touto metodou souvisí s parametrem `page` popisovaném níže.

Dále rozhraní `Search` definuje další dvě přetížené metody s jinou signaturou. Název metody je `search(...)` a může mít parametry:

- Povinný parametr `String query` reprezentuje dotaz předaný metodě k vyhledávání z knihovny Lucene.
- Nepovinný parametr `Set<String> subsidiaryWords` je množina pomocných slov, která upřesňují vyhledávaný dokument. Strategie sestavení dotazu je popsaná v sekci 4.2.

[4]UML doménový diagram byl vygenerován za pomoci doplňku pro Eclipse, který je dostupný na `http://www.objectaid.net/update`. Návod na jeho instalaci je pak na adrese `http://fuzz-box.blogspot.cz/2012/09/how-to-generate-uml-diagrams-from-java.html`.

- Povinný parametr `int page` určuje jaká část nalezených výsledků se má vrátit. Tento parametr nepřímo souvisí s nastavením max. velikostí množiny nalezených kurzů (označenou pro účely tohoto popisu písmenem **X**).

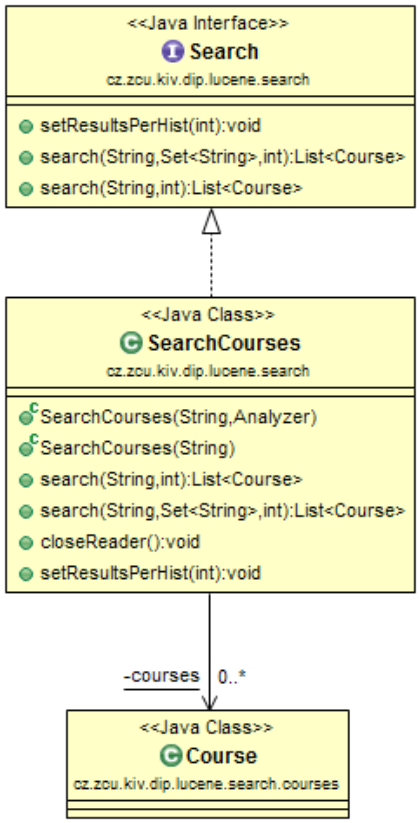
  Následuje vysvětlení významu tohoto parametru na příkladu. Nechť v indexu existuje 100 záznamů a z toho je 50 relevantních k zadanému výrazu. Nalezené odpovídající záznamy jsou ve výchozím nastavení seřazeny sestupně podle relevance. Pokud **X** je rovno deseti a hodnota parametru **page** je 2 znamená to, že se získá druhá desítka z padesáti relevantních záznamů. Pro hodnotu 0 vrátí celou množinu nalezených výsledků, tedy všech padesát záznamů.

Třída `SearchCourses`

Tato třída implementuje rozhraní `Search`. Obsahuje dva konstruktory. Jeden z konstruktorů umožňuje nastavit pro vyhledávání konkrétní analyzátor a druhý konstruktor bez parametru `analyzer` použije výchozí analyzátor `StemmAnalyzer`, viz sekce 3.4.6. Kromě pomocných metod v této třídě jsou důležité privátní metody `buildQuery(...)`, které analyzují předaný dotaz a v jednom případě přetížené metody, konkrétně s parametrem navíc `Set<String> subsidiaryWords`, přidá pomocná slova a fráze do dotazu přizpůsobeného knihovně Lucene. Pomocná slova jsou opatřena znakem označujícím fuzzy logiku a fráze stejným znakem, ale u frází má význam proximitní vlastnosti. Poslední zmíněnou úpravu provádí metoda `makeFuzzyOrProximitiQuery(String words)` s návratovou hodnotou modifikovaného řetězce předaného metodě jako parametr.

Třída `courses.Course`

V této třídě se atributy ve své většině shodují s možnými elementy kurzu, které jsou též uvedeny v příloze a v tab. A.2 a A.4. Jedinou výjimkou je element *originalID*, který se v upravené databázi již nenachází a je složen ze dvou jiných elementů a to z elementu `courseNumber` a `deparment`. Pro všechny atributy třídy jsou také setry a getry. Účelem této třídy je uchovávat všechny možné informace o kurzu nacházejícím se v databázi.



Obrázek 5.5: Doménový model knihovny SearchCourses.

5.3 Knihovna `IndexCoursesXML`

Tato knihovna byla vytvořena na platformě Java SE. Účelem této knihovny je indexace XML databáze za pomoci knihovny Lucene. Na obr. 5.6 umístěném na konci této sekce je uveden doménový model. Všechny třídy, rozhraní a další mají základní prefix `cz.zcu.kiv.dip.lucene.index` a proto budou popisovány bez tohoto prefixu. Dále je uvedena alternativa uložení XML dokumentu se zdrojovými daty a potom následuje popis této knihovny.

5.3.1 Alternativa k XML dokumentu

Proces vytvoření indexu pomocí knihovny Lucene by mohl také být proveden nad daty uloženými v XML databázi (nativní XML databáze, která rozumí

vnitřní struktuře XML dokumentu a obsahuje veškeré funkce pro manipulaci s těmito daty) namísto od XML dokumentu. Ovšem vytvoření takovéto XML databáze by bylo neefektivní, jelikož se nepředpokládá, že se indexace bude provádět často a přístup k datům v původní XML struktuře není vůbec předpokládán.

Rozhraní `Index`

Rozhraní `Index` definuje čtyři přetížené metody s jinou signaturou. Název metody je `indexCollection(...)` a může mít parametry:

- Povinný parametr `String pathToIndex` slouží k nastavení absolutní nebo relativní cesty ke složce, která je určena k uložení vytvořeného indexu.
- Povinný parametr `String sourceXMLCollection` představuje absolutní nebo relativní cestu k XML dokumentu s daty.
- Nepovinný parametr `String outStatistics` udává absolutní nebo relativní cestu k výstupnímu XML souboru se statistiky o databázi.
- Nepovinný parametr `Analyzer analyzer` definuje použitý analyzátor k indexaci, který je třídou v knihovně Lucene viz sekce 3.4.

Třída `IndexCoursesXML`

Tato třída implementuje rozhraní `Index` a její úkol je naindexovat XML dokument. Tato třída provádí indexaci pomocí analyzátoru předaný metodě `indexCollection(...)` popsanou výše. Pokud není předán žádný analyzátor zmíněné metodě je použit výchozí - `StemmAnalyzer`, viz sekce 3.4.6. Při procesu indexace je nutné analyzovat zároveň XML dokument, k čemuž se využívá knihovna `Digester` viz sekce 3.8. Způsob jakým se vytváří index je popsán v kapitole 4. Třída `IndexCoursesXML` obsahuje instanci třídy `utils.StatisticsCollection` k zaznamenávání statistik o datech v dokumentu a další instance třídy `writer.writeStatistics`, jejímž prostřednictvím se zapisují získané statistiky během zpracování dokumentu.

Třída `utils.StatisticsCollection`

Úkolem této třídy je sbírat statistiky. Obsahuje dvě hlavní struktury ke shromažďování dat. První struktura uchovává záznamy jednotlivých let, ve kterých byly kurzy nabízeny a odpovídají hodnotám v elementu *yearOffered* (tzn. hodnoty 2012, 2011, ...). Ve druhé struktuře jsou uchovávány objekty `utils.University`.

Obsahuje metody pro přidání dat do obou struktur, kterými jsou `addInfoUniversity(Course course)` a `addToYears(String yearsOffered)`. Dále obsahuje getry k získání obou struktur a další méně důležité metody.

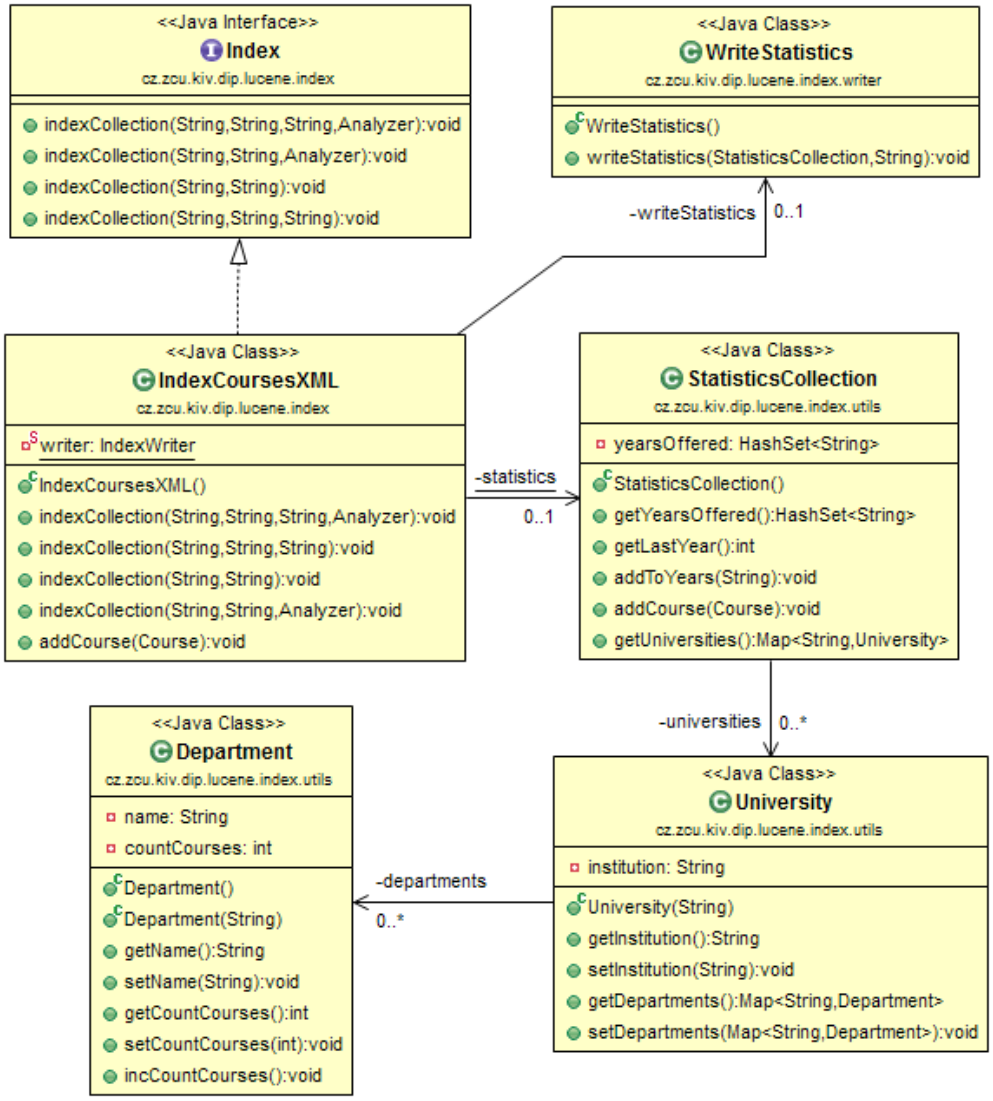
Třída `writer.WriteStatistics`

Třída `writer.WriteStatistics` slouží k zapsání nahromaděných statistik v objektu třídy `utils.StatisticsCollection` do XML souboru. Struktura výsledného XML dokumentu má kořenový element *statistics*. V kořenovém elementu mohou být jen dva typy elementů, kterými jsou: *university* a *yearoffered*. Element *university* obsahuje element *department* s výskytem 1 a více, dále obsahuje po jednom výskytu elementy *name* (jež obsahuje název univerzity) a *countcourses* (obsahem je počet nabízených kurzů danou univerzitou). V elementu *yearoffered* se vyskytuje element *year* s četností 0 a více a s obsahem udávající rok. Jednotlivá léta odpovídají hodnotám v elementu `yearOffered` v indexovaném XML dokumentu. Následuje příklad takového XML dokumentu se statistiky.

```
1  <statistics>
2    <university>
3      <name>
4        The Juilliard School
5      </name>
6      <countcourses>
7        758
8      </countcourses>
9      <department>
10       FLHMU
11     </department>
12   </university>
13   <yearoffered>
14     <year>
15       2008
16     </year>
17   </yearoffered>
18 </statistics>
```

Třídy `Course`, `University` a `Department`

Instance třídy `utils.University` obsahuje informace o univerzitě jako její název a seznam všech kateder a další třída `utils.Department` obsahuje informace o katedře (název katedry a počet nabízených kurzů). Třída `courses.Course` je shodná se stejnojmennou třídou v knihovně `SearchCourses` viz sekce 5.2.



Obrázek 5.6: Doménový model knihovny IndexCoursesXML.

6 Ověření funkčnosti aplikace

Vytvořená webová aplikace byla spuštěna na Notebooku - Intel(R) Celeron(R) CPU B815 1.60GHz, RAM 4GB s OS Windows 7 Home Premium. V následujícím textu bude diskutována relevantnost vyhledávaných kurzů vytvořené aplikace.

6.1 Porovnání výsledků

Zde bude uvedeno několik výsledků hledání se stejně nastavenými parametry, ale s různou strategií hledání. Všechny dotazy byly zadané do pole *Keywords and phrases:* viz uživatelský manuál 7. Rozšířit hledaný výraz či frázi je možné s využitím ACM CCS anebo s nástrojem WordNet, podrobné vysvětlení významu a použití těchto rozšíření je uvedeno v sekci 4.2.

Aby bylo možné porovnat všechny způsoby vyhledávání je použit jeden hledaný výraz, a postupně jsou na něj aplikovány všechny varianty strategie. Zvoleným výrazem pro vyhledávání v obsahu kurzu je slovo **network**. Následuje výpis třech výsledků hledání s nastavenou různou strategií rozšíření dotazu s tím, že je v příkladech uvedeno omezené množství informací o kurzu. Uvedenými informacemi jsou: název univerzity, kurzu, katedra a obsah sylabu daného kurzu. Každý uvedený výsledek hledání byl vyhodnocen knihovnou Lucene jako nejrelevantnější pro danou variantu, byl tedy na prvním místě mezi výsledky hledání. Vysvětlující popis k ukázkám následuje po uvedených příkladech.

1. **Bez rozšíření**

   Institution: *Arkansas Tech University*
   Title: *HETEROGENEOUS NETWORKS*
   Department: *INFT*

   Networking in a heterogeneous environment.

2. **Rozšíření s WordNet**

   Institution: *University of Michigan-Flint*
   Title: *Computer Networks II.*
   Department: *CSC*

Practical, advanced concepts in computer **networking** that extend the theoretical knowledge gained in CSC 335. The common gateway interface, **network** security and **network** monitoring, scripting and programming languages for computer **networks**, electronic commerce techniques, **web** graphics, virtual private **networks**, construction and administration of internet servers, and the interface to **network**-connected databases.

3. **Rozšíření s ACM CCS**

   Institution: *George Mason University*
   Title: *Distributed Software Engineering*
   Department: *SWE*

   Hands-on introduction to techniques and programming interfaces for distributed software engineering. **Networking protocols** at several layers. Construction of distributed and concurrent software using **network protocol services**. Applications of Internet and web-based software.

Ve všech příkladech jsou tučným písmem zvýrazněny původní hledané výrazy a také pomocné pojmy. První příklad je výsledek hledání odpovídající původnímu výrazu, který nebyl žádným způsobem rozšiřován. Dotaz tedy měl tvar `+overview:network`[1].

Ve druhém příkladu bylo využito rozšiřování pomocí nástroje WordNet. V tomto případě pro slovo *network*[2] WordNet vygeneroval varianty:

- electronic network,
- web,
- meshwork,
- meshing,
- mesh.

Z porovnání pomocných slov vygenerovaných nástrojem WordNet a s obsahem druhého příkladu je zjevné, že výsledek byl ovlivněn z hlediska pořadí

[1] Syntaxe dotazu je popsána v sekci o Apache Lucene viz 3.4.5.
[2] Slovo *networks* bylo převedeno lemmatizátorem na *network*.

podle relevantnosti (danému dokumentu se zvýšilo skóre během vyhledávání v indexu) pomocným slovem *web*. Vygenerované fráze a slova jsou ve vyhledávání nepovinná a výsledný dotaz má tvar:

```
+overview:network overview:"electronic network"∽2
overview:web∽2 ...
```

V dalším a posledním příkladu byl základní dotaz rozšířen o pojmy z ACM CCS. I v tomto příkladě byl pojem *networks* lemmatizován a nalezené pojmy z ACM CCS jsou:

- network algorithms, network protocols, network components, network types, network performance evaluation, network architectures, network properties, network services.

Pokud se porovnají pomocná slova a fráze získané pomocí ACM CCS s obsahem sylabu ve třetím příkladu vyjde najevo, že výsledek byl ovlivněn pomocnými frázemi **network protocols** a **network services**, obdobně jako u druhého příkladu slovem **web**. Vzhledem k nalezeným pojmům bude mít výsledný dotaz tvar:

```
+overview:network overview:"network algorithms"∽2
overview:"network protocols"∽2 ...
```

Počet nalezených kurzů odpovídajících každé variantě popsané výše bylo samozřejmě více a navzájem se všechny mezi sebou lišily (porovnáno bylo celkem pět výsledků hledání, počítáno i se sloučenými kurzy). O všech výsledcích hledání je oprávněné tvrdit, že jsou relevantní. Ovšem už není možné s jistotou tvrdit, že některé výsledky jsou výrazně relevantnější než jiné a v některých případech jde spíše o subjektivní názor. Tato neurčitost je mimo jiné způsobena hlavně nedostatečnou znalostí databáze. Potřebná rešerše databáze by byla nad rámec této práce. Při porovnávání na jiné množině hledaných slov je patrné, že pokud by tazatele zajímaly například počítačové vědy, najde pomocí rozšíření dotazu více relevantních kurzů. Kurzy navíc častěji mají popis s počtem slov převyšující hodnotu dvacet.

Další porovnání

Jiným hledaným výrazem je slovo **hardware**. Z výsledků hledání uvedených níže je možné vypozorovat markantnější rozdíl než u předešlého příkladu. Opět následuje příklad třech výsledků považovaných za nejrelevantnější pro každou variantu hledání s rozšířením dotazu. Nastavené parametry hledání jsou shodné jako u předchozích příkladů.

1. **Bez rozšíření**
   Institution: *University of North Texas*
   Title: *Foundations for Communication Design*
   Department: *ADES*

   Computer **hardware** and software and their application in communication design.

2. **Rozšíření s WordNet**
   Institution: *University of North Texas*
   Title: *Foundations for Communication Design*
   Department: *ADES*

   Computer hardware and software and their application in communication design.

3. **Rozšíření s ACM CCS**
   Institution: *Purdue University*
   Title: *Electrn Systm Fabrictn*
   Department: *ECET*

   Credit Hours : 2.00. The course includes electronic schematic, printed circuit board design and fabrication using **Electronic Design Automation** (EDA) tools. Designing electronic circuit schematic, schematic annotation, netlist file generations, electronic packaging selection, printed circuit board (PCB) artwork design using auto router and manual router software tools. Populate the printed circuit board with electronic components ; solder using hand tools and testing/debug the electronic **hardware** into an operational system using bench-top instruments. Course teaches prototyping electronic projects. Typically offered Fall Spring.

Z těchto výsledků je patrné, že v tomto případě nemělo rozšíření dotazu pomocí nástroje WordNet žádný efekt na první nalezený kurz. Ostatní kurzy

se již ve své většině lišily. Rozšíření dotazu pomocí ACM CCS naopak od předchozích příkladů ovlivnilo výsledek hledání nejvíce. Tento fakt lze odůvodnit tím, že použitý klasifikační systém je zaměřen na počítačové vědy na rozdíl od WordNet a tak nabídne více relevantní alternativy k původnímu výrazu. To vše za předpokladu, že při dotazu ve tvaru *hardware* má tazatel zájem o oblast počítačových věd.

Dále bylo vyzkoušeno několik dalších variant hledání. Účinnost rozšiřování dotazu klesá s rostoucím počtem hledaných pojmů. Například k rozšíření dotazu pomocí ACM CCS dojde nejčastěji, pokud je dotaz dvouslovný (bez uvozovek), což je dané obsahem klasifikačního systému. Ovšem pokud dojde k tomuto rozšíření je výsledný dotaz pravděpodobně výstižnější, než kdyby došlo k rozšíření nástrojem WordNet. Příkladem může být výše uvedený dotaz (*hardware*).

U tří a víceslovného dotazu ve většině případů dojde k rozšíření nástrojem WordNet, což může rozšířit dotaz o slova a fráze, která jsou významem od sebe příliš vzdálená. Pokud k takovému rozšíření dojde, může to mít za následek znegování všech pomocných pojmů tzn. výsledný efekt bude srovnatelný, jako kdyby k rozšíření nedošlo. Vedlejším účelem rozšiřování dotazu je zobrazování možných alternativ, které tazatel v příštím hledání může použít a tím zpřesnit oblast zájmu.

6.2 Porovnání různých analyzátorů

V následujícím textu bude popsán rozdíl mezi výsledky nalezenými pomocí knihovny `SearchCourses` využívající v jednom případě analyzátor `StemmAnalyzer` a ve druhém případě standardní analyzátor `StandardAnalyzer`. Proces indexace dat pomocí knihovny `IndexCoursesXML` a následné vyhledávání kurzů v indexu bylo provedeno za pomoci shodných analyzátorů pro oba procesy. Parametry vyhledávání jsou stejné jako v sekci 6.1 a stejně tak i hledaná slova **networks** a **hardware**. Výpisy vyhledaných výsledků za pomoci standardního analyzátoru pro slovo **networks** jsou v příloze A.2.

Z porovnání obou výsledků vyhledávání pro klíčové slovo **networks** je zřejmé, že výsledek byl ovlivněn procesem stemmatizace realizovaný analyzátorem `StemmAnalyzer`. Standardní analyzátor nezaznamenal shodu při vyhledávání mezi slovy se stejným kmenem jako byla hledaná slova, jeikož tento analyzátor postrádá funkci stemmatizace.

Pro vyhledávaný výraz **hardware** za pomoci standardního analyzátoru se první tři výsledky nelišily od výsledků vyhledaných za pomoci `StemmAnalyzer`.

V této fázi testování funkčnosti aplikace není možné jednoznačně vybrat vhodnější analyzátor pro vyhledávání, jelikož ohodnocení relevance výsledků záleží na subjektivním posouzení tazatele. Proto by bylo vhodné v rámci rozšíření této práce nasadit aplikaci do provozu a zaznamenávat reakce uživatelů na výsledky hledání poskytnutými za pomoci různých analyzátorů. Na základě získaní dostatečného množství reakcí od uživatelů by se rozhodlo o vhodnějším analyzátoru. Nicméně výchozím analyzátorem pro knihovny `IndexCoursesXML` a `SearchCourses` byl zvolen `StemmAnalyzer`. Volba tohoto analyzátoru je odůvodnitelná faktem, že pravděpodobnost shody hledaného slova se slovy uloženými v indexu je větší po aplikování procesu stemmatizace.

Jiné možnosti rozšíření dotazu

Rozšíření dotazu je reálné i s použitím jiných prostředků, zejména alternativy popsané v sekci 3.3. Vhodným rozšířením podle získaných zkušeností by bylo zařazení klasifikačních systémů nebo slovníků zaměřených na jednotlivé oblasti, které by nebyly příliš rozsáhlé a odpovídaly by členitostí a rozsahem ACM CCS. Pokud by se použilo dostatečné množství prostředků pokrývajících většinu nabízených oborů na univerzitách, které se nacházejí v databázi, mohl by se vynechat začleněný tezaurus WordNet. Použitý tezaurus totiž nabízí k jednotlivým výrazům příliš obecné pojmy.

Pokud by se dostatečně různé oblasti oborů pokryly vybranými ontologiemi, slovníky apod. mohla by se ve výsledné aplikaci implementovat funkce umožňující výběr zaměření na konkrétní oblast zájmu.

7 Uživatelská dokumentace

Obsahem této kapitoly je uživatelská dokumentace popisující všechny možné akce a činnosti, které lze provádět jako běžný návštěvník webových stránek. Některé obrázky v této kapitole záměrně zobrazují jen určité části stránek. Po instalaci aplikace je nutné pro její správné fungování nastavit několik parametrů viz sekce 7.3.

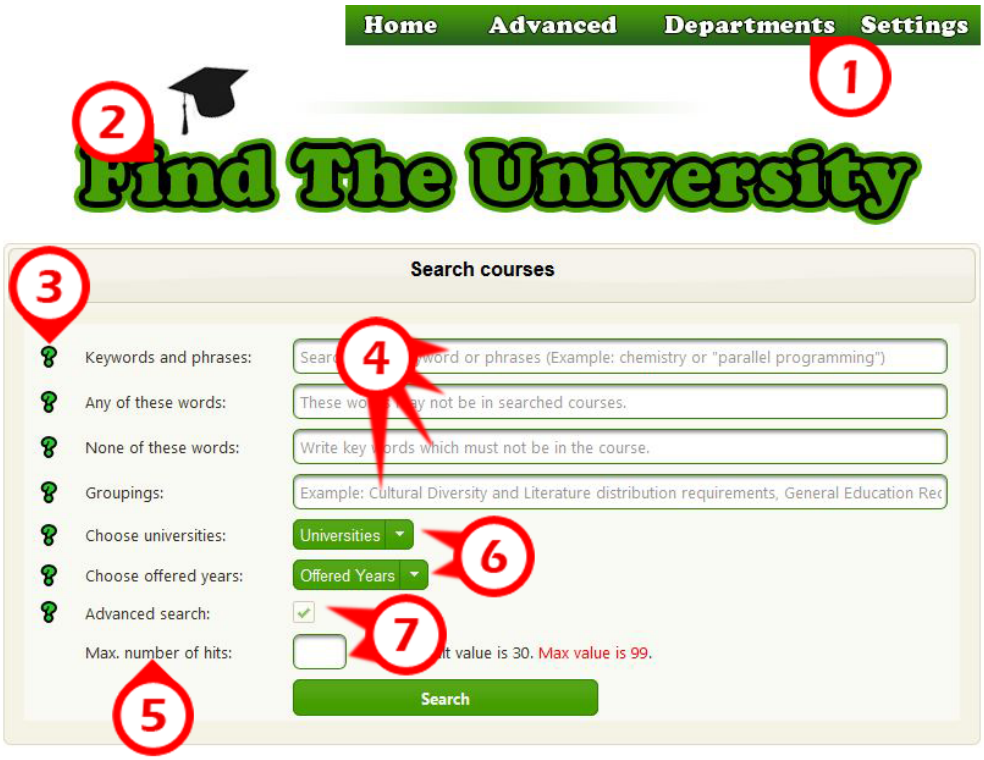
Vytvořit index z XML dokumentu lze pomocí stránky popisovanou v sekci 7.3. Proces vytváření indexu trvá na notebooku s Intel(R) Celeron(R) CPU B815, 1.60GHz, RAM 4GB přibližně 3 minuty[1]. Aplikace umožňuje indexovat jen po jednom XML dokumentu a pro účely vyhledávání je možné využívat jen jeden index, který je nastaven.

V následujícím textu bude ke každé webové stránce (rozdělené do sekcí) uveden otisk obrazovky s označenými částmi s čísly a poté bude následovat příslušný popis obrázku. Očíslované výčty odpovídají číslování jednotlivých částí stránky. Pokud je v této kapitole diskutováno o „hledaných slovech“, pak to zahrnuje také pojem „hledaných frázích“. Mapa webových stránek je umístěna v sekci 5.1.2.

7.1 Domovská stránka

Domovská stránka na níže uvedeném obrázku nabízí základní formulář (tabulku) pro vyhledávání v kurzech a v horní části je navigace vedoucí na všechny ostatní stránky, kromě stránek zobrazujících výsledky. Vyhledávání započne po potvrzení tlačítkem **Search**.

[1] Jako zdrojový XML dokument pro indexaci byl použit redukovaný XML dokument vzniklý transformací z původních dokumentů, viz kapitola 2.



Obrázek 7.1: Domovská strának webu.

1. Navigace. Stránky příslušné jednotlivým odkazům v navigaci jsou popsané v dalších sekcích.

2. Logo webové aplikace.

3. Sloupec otazníků. Jednotlivé otazníky po přejetí ukazatelem myši nad obrázkem se zobrazí nápověda k danému řádku formuláře.

4. Pole pro zadání vyhledávaných klíčových slov.

   - **Keywords and phrases** je pole pro zadání povinných slov a frází[2], které se **musejí vyskytovat** v obsahu sylabu v porovnávaném kurzu[3].
   - **Any of these words** je pole pro zadání nepovinných slov a frází, které se vyskytují v obsahu sylabu porovnávaného kurzu.

[2]Fráze musejí být napsány v uvozovkách.
[3]V databázi kurzů se jedná o obsah elementu *overview*

- **None of these words** je pole pro zadání zakázaných slov a frází, které se **nesmějí vyskytovat** v obsahu sylabu porovnávaného kurzu.
- **Groupings** je pole, které nemá v kurzech pevně stanoven obsah. Nicméně po jednoduché analýze je zřejmé, že obsahem je krátký popis zařazující daný kurz do různých skupin, například do skupiny humanitních věd nebo technického zaměření kurzu atd.

5. Sloupec s názvy pro každé kritérium hledání na dané řádce.

6. Upřesňující kritéria:

   - kritériem nazvaným **Choose universities** se mohou zvolit univerzity, ve kterých se má hledat,
   - kritériem nazvaným **Choose offered years** se mohou upřesnit roky, po které byl hledaný kurz nabízen. Toto kritérium se využije především na základě zkušeností z minulých hledání.

7. „Způsob“ hledání a počet výsledků:

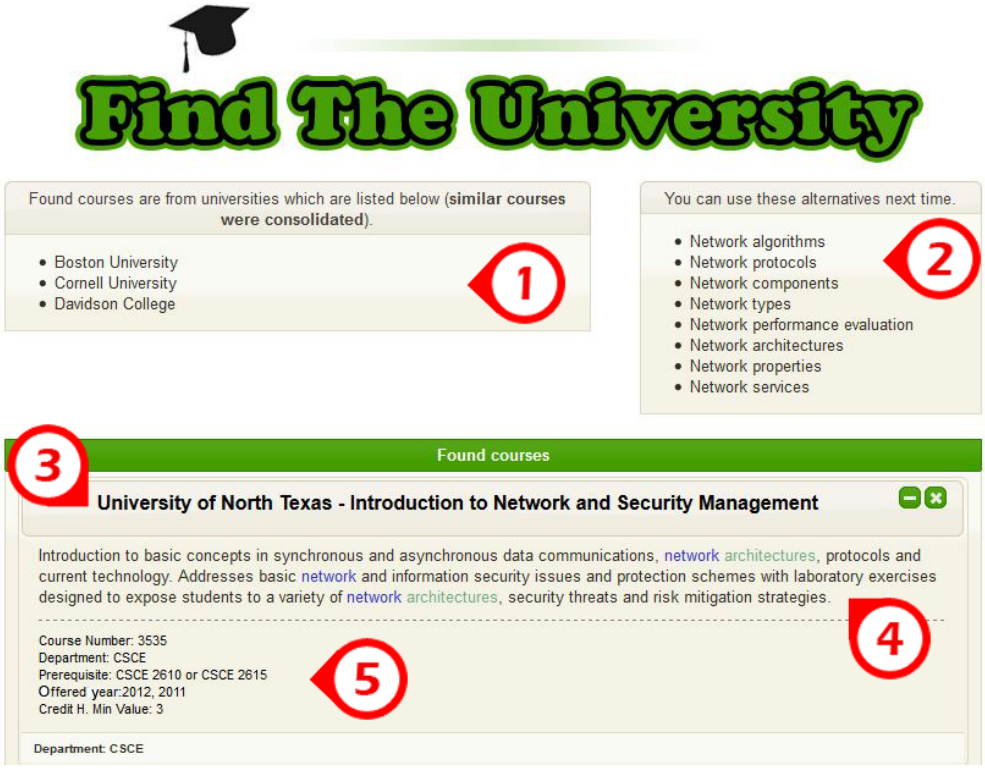
   - kritérium **Advanced search** umožňuje zapnout funkci pro rozšíření dotazu zadaného do pole **Keywords and phrases** a tím upřesňovat hledanou množinu kurzů[4],
   - kritérium **Max. number of hits** nabízí zvolit počet vyhledaných kurzů s největší relevantností vzhledem k hledaným kritériím.

Stránka s pokročilým vyhledáváním obsahuje od této stránky navíc část pro zvolení číselných rozsahů daných parametrů kurzu. Dále část pro zadání zbylých kritérií, které kurz může obsahovat.

7.2 Stránka s výsledky - boxy

Na obr. 7.2 je zobrazena stránka s vyhledanými kurzy (počet hledaných kurzů byl omezen na jeden). Na tuto stránku je uživatel přesměrován z domovské stránky vyvoláním akce hledání tlačítkem **Search**. Z této stránky si lze volbou *View 2* z navigačního menu zobrazit výsledky také ve formě tabulky popsané v sekci 7.5 a naopak.

[4]Rozšíření dotazu je popsáno v kapitole 4.



Obrázek 7.2: Stránka se zobrazením výsledků roztříděných do boxů.

1. Hledané kurzy musely příslušet právě jedné z uvedených univerzit v této tabulce označenou číslem 1. Takže byly upřednostněny kurzy z těchto uvedených univerzit, které se zvolily před hledáním na domovské stránce viz sekce 7.1.

2. Tento výčet zobrazuje pomocné pojmy ke slovům zadaným do pole **Keywords and phrases** na domovské stránce. Pro zapnutí speciálního režimu rozšiřování dotazu je nutné mít povolené kritérium **Advanced search** nacházející se také na domovské stránce.

3. Název univerzity společně s názvem hledaného kurzu.

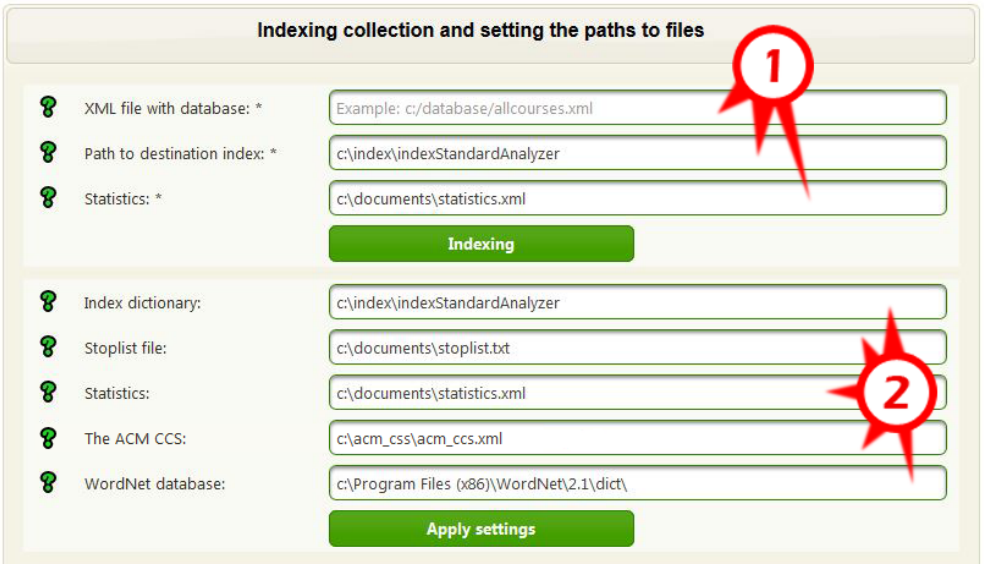
4. Obsah sylabu nalezeného kurzu. Hledaná povinná slova jsou zvýrazněna modrou barvou a pomocná slova, viz výčet v části č. 2, jsou zvýrazněna lehce zelenou barvou.

5. V této části jsou uvedeny ostatní detaily o nalezeném kurzu. Pokud bylo nalezeno více podobných kurzů se společným sylabem, názvem kurzu a univerzity jsou tyto kurzy sloučeny do jednoho. V detailech o kurzu

se to projeví tím, že například u detailu *Offered year* se zobrazí více dat.

7.3 Stránka s nastavením aplikace

Na následujícím obr. 7.3 je stránka s formulářem (tabulkou) s možností nastavení parametrů pro proces indexace XML dokumentu s daty o kurzech[5] a další formulář k nastavení cest důležitých pro fungování aplikace.



Obrázek 7.3: Formulář pro nastavení parametrů aplikace.

1. V první části stránky lze nastavit parametry pro indexaci:
   - pole **XML file with database *** slouží k nastavení cesty k XML dokumentu obsahující data o kurzech,
   - pole **Path to destination index *** slouží k nastavení cesty vedoucí k místu kam se uloží vytvořený index z XML dokumentu,
   - do pole **Statistic *** se zadá cesta do složky kam se uloží vytvořené statistiky během průběhu indexace XML dokumentu.

[5]Tato struktura XML dat musí být validní oproti XSD souboru s názvem *allcourses_reduced.xsd*. Viz příloha A.1.

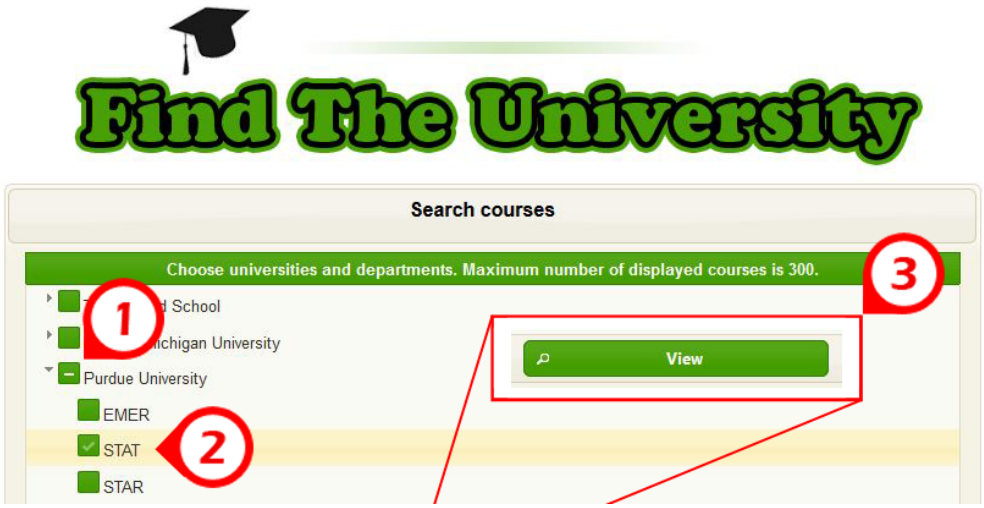
2. Druhá část slouží k nastavení cest ke zdrojovým datům nutným pro správné fungování a k plnohodnotnému využití potenciálu aplikace.

   - Obsahem pole **Index dictionary** je cesta vedoucí k již existujícímu indexu anebo k indexu vytvořeného v rámci první části.
   - Pole **Stoplist file** slouží k nastavení cesty k textovému souboru se seznamem stop slov[6].
   - Cesta nastavená v poli **Statistics** vede k XML dokumentu se statistikami o databázi dat, viz pole **XML file with database: ***.
   - Pole **ACM CCS** je určené pro nastavení cesty k XML dokumentu s ACM CCS[7].
   - Obsahem pole **WordNet database** je cesta vedoucí k nainstalované WordNet databázi[8].

7.4 Stránka s výběrem kurzů podle kateder

Stránka s možností filtrovaní kurzů podle univerzit a kateder je částečně zobrazena na obr. 7.4, po kterém následuje příslušný popis. Tato funkčnost vybrání kurzů podle kateder je doplňujícím rozšířením. K této stránce uživatel přistoupí, pokud již má nějaké zkušenosti a znalosti ze základního hledání. Takže například, když se uživatel zajímá o kurzy spadající pod stejnou katedru si může zobrazit všechny kurzy z dané katedry.

[6]Definici stop slov a souvislost s touto aplikací je uvedena na začátku kapitoly 3.
[7]Použitý a otestovaný klasifikační systém je na CD přiloženém k této práci.
[8]WordNet databáze je popsaná v sekci 3.2.



Obrázek 7.4: Stránka s možností výběru kurzů podle kateder.

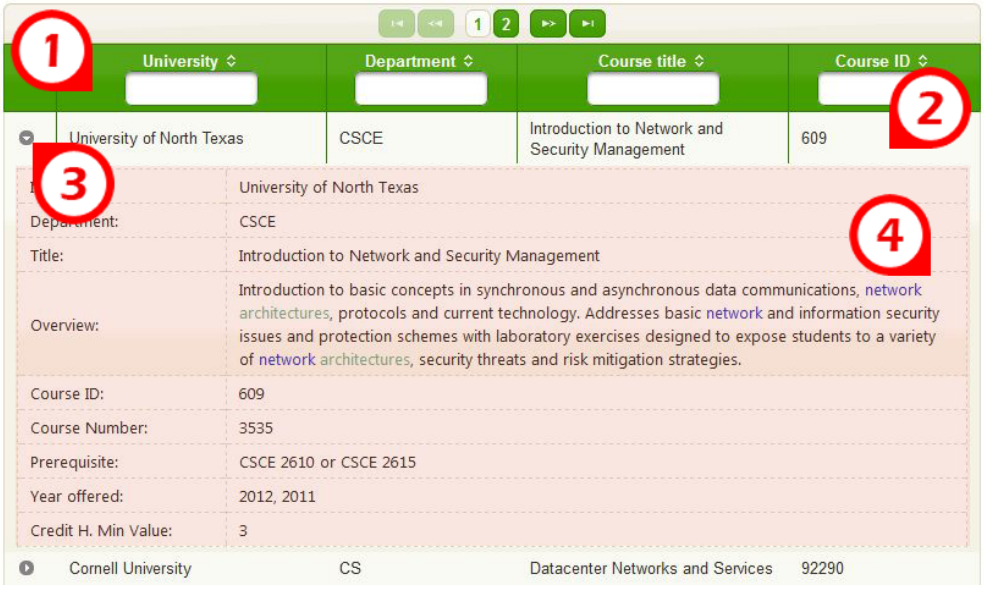
1. První označení zobrazuje možnost rozvinutí nabídky kateder, viz další bod. Je možné zaškrtnutím vybrat všechny katedry na dané univerzitě.

2. Toto označení znázorňuje možnost výběru z řad kateder příslušné univerzitě ve stromové struktuře.

3. Jelikož se do otisku obrazovky nevešlo vše, je tlačítko **View** potvrzující výběr a následné zobrazení kurzů „vytáhnuto“ nahoru, aby bylo zobrazeno na ukázce.

7.5 Stránka s výsledky - tabulka

Po úspěšném nalezení vyhledávaných kurzů je uživatel přesměrován na tuto stránku z domovské stránky anebo ze stránky *department.xhtml* , viz sekce 7.4. Na následujícím obrázku je ve formě tabulky s možností rozvinutí detailu jen část z celé stránky zobrazující nalezené kurzy. Z této stránky lze volbou *View 1* z navigačního menu zobrazit výsledky ve formě tabulky popsané v sekci 7.6, pokud se vyhledávaly kurzy podle katedry z výchozí stránky popsané v sekci 7.4. V případě vyhledávání kurzů ze stránek popsaných v sekci 7.1 se volbou *View 1* zobrazí tabulka s možností náhledu na detail kurzu přes dialog, viz sekce 7.2.

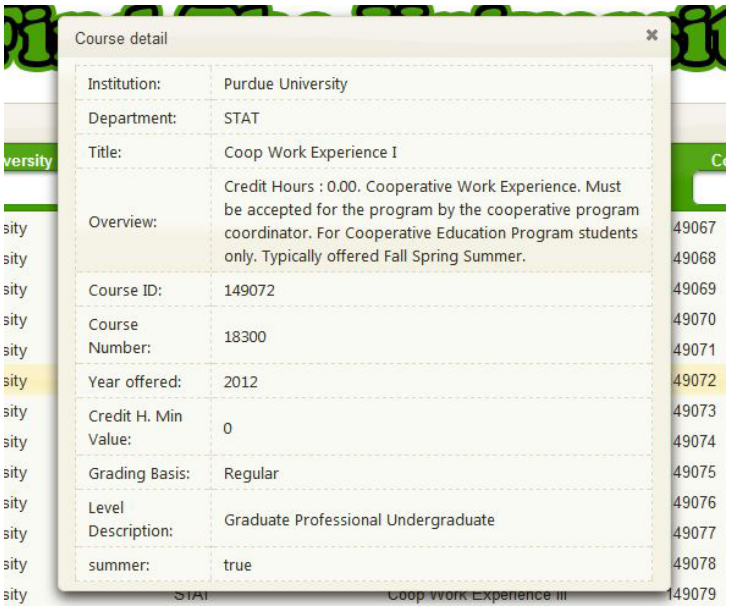


Obrázek 7.5: Výsledky zobrazené ve formě tabulky a s možností detailu.

1. První část tabulky umožňuje filtrovat kurzy podle názvu univerzity, kurzu, katedry a identifikačního čísla kurzu.
2. Řádek tabulky s výpisem jen některých informací o kurzu.
3. Tlačítko s funkcí rozvinutí detailu o kurzu, viz následující bod.
4. Oblast zvýrazněna nádechem do růžova obsahuje všechny dostupné detaily o kurzu.

7.6 Stránka s výsledky - dialog

Přesměrování na tuto stránku vede ze stránky *department.xhtml* po úspěšném nalezení skupiny kurzů podle kateder, viz sekce 7.4. Výsledky jsou zobrazeny ve formě tabulky stejně jako v sekci 7.5, ale s rozdílem způsobu zobrazení detailu kurzu. V tomto případě se po kliknutí na řádek/položku v tabulce zobrazí dialog s detailem. Možnost rozvinutí detailu jako v předchozím případě zde není nutná. Na následujícím obrázku je pohled na zmíněný dialog. Z této stránky si lze volbou *View 1* z navigačního menu zobrazit výsledky také ve formě tabulky popsané v sekci 7.5 a opět přepnout zpět na tento dialog.



Obrázek 7.6: Detail kurzu zobrazený pomocí Primefaces komponenty *dialog*.

8 Závěr

Nejprve byla prozkoumána struktura dat poskytnutá ve formě XML dokumentů obsahující data o kurzech nabízených univerzitami. Jelikož původní data obsahují příliš mnoho nadbytečných informací, než je potřeba pro účely této práce, byla provedena transformace dat pomocí XSLT procesoru, která odstranila nadbytečné informace. K původním i redukovaným datům byla vytvořena XSD definice. Redukcí dat došlo ke značnému zmenšení velikosti původních XML dokumentů, což zmenšilo potřebný výpočetní výkon na jejich další zpracování a ulehčilo tím práci. Popis struktury dat a navrhnuté transformace je v kapitole 2.

Před samotným vývojem aplikace byly prozkoumány existující nástroje a jejich alternativy, s jejichž pomocí společně s analýzou dat se navrhla strategie indexace XML dokumentů a rozšiřování dotazu. Rozšiřování dotazu spočívá v přidávání příbuzných pojmů k dotazu, čímž se upřesňuje oblast zájmu hledaných kurzů. Strategie je navrhnuta takovým způsobem, aby se dosahovalo co možno nejvíce relevantních výsledků ve vyhledávání. Dále byly zkoumány různé technologie, nástroje, knihovny atd. potřebné k dosažení úspěšné implementace cílové aplikace společně s odůvodněním jejich výběru. Při výběru technologií byl kladen důraz na jednoduchost, dostupnost a na současné moderní trendy. Použité nástroje a navrhnutá strategie k indexaci dat a následného vyhledávání v indexu jsou popsány v kapitolách 3 a 4.

V rámci této diplomové práce byly vytvořeny dvě knihovny využívající knihovnu Lucene z nichž jedna je určená pro vytvoření indexu a druhá k následnému vyhledávání v tomto indexu. Webová aplikace využívá vytvořené knihoveny a poskytuje funkčnost potřebnou pro vyhledávání kurzů společně s různými možnostmi zobrazení výsledků. Při vyhledávání je možné zapnout funkci rozšiřování dotazu o příbuzné pojmy. Bylo vytvořeno grafické rozhraní pro pohodlné nastavení cest ke zdrojovým datům společně s možností vytvoření indexu z XML dokumentu s daty o kurzech. Webová aplikace je popsána v kapitole 7 a návrh její implementace je v kapitole 5.

Jako možné rozšíření výsledné aplikace se nabízí zavedení uživatelských rolí, zejména pak role *administrátor*. Role administrátora by spočívala především v nastavování aplikace, jež je v současné aplikaci dostupná běžnému uživateli díky absenci těchto rolí. Existence rolí nebyla od aplikace vyžadována. Dalším rozšíření by mohlo být zdokonalení procesu napomáhání uživateli najít relevantní kurzy.

Seznam obrázků

Seznam tabulek

Seznam zkratek

API	Application Programming Interface
XML	Extensible Markup Language
XSD	XML schéma, které definuje soustavu pravidel a specifikací.
XSLT	eXtensible Stylesheet Language Transformations
ACM	Association for Computing Machinery
HTML	HyperText Markup Language
JSF	JavaServer Faces
CDK	Component Development Kit
JSP	Java Server Pages
EJB	Enterprise JavaBeans
SAX	Simple API for XML
DOM	Document Object Model
MVC	Model-view-controller
CSS	Cascading Style Sheets
CCS	The ACM Computing Classification System

Literatura

[Acm(2014)] [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://dl.acm.org/ccs.cfm`.

[Ant(2014)] *Apache Ant 1.9.4 Manual* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://ant.apache.org/manual/index.html`.

[Bnf(2012)] *Class QueryParser* [online]. 2012. [cit. 9.5.2014]. Dostupné z: `http://lucene.apache.org/core/4_0_0/queryparser/`.

[Dig(2014)] *Learning Apache Commons Digester 3* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://docs.oracle.com/javaee/1.4/tutorial/doc/JAXPDOM2.html`.

[Dom(2011)] *When to Use Dom* [online]. 2011. [cit. 9.5.2014]. Dostupné z: `http://ted-gao.blogspot.cz/2011/09/apache-commons-digester-3-basics.html`.

[Dot(2013)] *Apache Lucene - Query Parser Syntax* [online]. 2013. [cit. 9.5.2014]. Dostupné z: `http://lucene.apache.org/core/2_9_4/queryparsersyntax.html`.

[Gla(2013)] *GlassFish vs Tomcat* [online]. 2013. [cit. 9.5.2014]. Dostupné z: `http://www.wikivs.com/wiki/GlassFish_vs_Tomcat`.

[Jet(2013)] *Why Use Jetty?* [online]. 2013. [cit. 9.5.2014]. Dostupné z: `https://www.webtide.com/choose/jetty.jsp`.

[Mav(2014)] *Welcome to Apache Maven* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://maven.apache.org/`.

[Mvc(2014)] *MVC Pattern* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://docs.oracle.com/javaee/1.4/tutorial/doc/JAXPDOM2.html`.

[Sax(2014)] *When to Use SAX* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://docs.oracle.com/javaee/1.4/tutorial/doc/JAXPSAX2.html`.

[Tom(2014)] *Apache Tomcat* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://tomcat.apache.org/`.

[Wor(2014)] *What is WordNet?* [online]. 2014. [cit. 9.5.2014]. Dostupné z: `http://wordnet.princeton.edu/`.

[Marchioni(2012)] MARCHIONI, F. *PrimeFaces vs RichFaces vs IceFaces* [online]. 2012. [cit. 9.5.2014]. Dostupné z: `http://www.mastertheboss.com/richfaces/primefaces-vs-richfaces-vs-icefaces`.

[Michael McCandless(2010)] MICHAEL MCCANDLESS, E. H. a. O. G. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Greenwich, CT, USA : Manning Publications Co., 2010. ISBN 1933988177, 9781933988177.

[Michal(2006)] MICHAL, P. K. a. J. *Dokumentografické informační systémy*. Karolinum, 2006. ISBN 8024611481.

A Přílohy

A.1 Struktura CD

Přiložené CD obsahuje následující adresáře:

- `soubory-xsd-xml`: Obsahuje všechny soubory ve formátu XSD a XML. Některé tyto soubory jsou také popsány v kapitole 2. K popisovaným XSD souborům existují stejnojmené XML dokumenty v témže adresáři. Ovšem chybějícími dokumenty s daty o kurzech jsou: `allcourses_reduced.xml` a `allcourses.xml`. Chybějící dokumenty jsou k dispozici na vyžádání od vedoucího této diplomové práce.
  - `allcourses_reduced.xsd` - Definuje strukturu XML dokumentu s redukovanými daty o kurzech.
  - `allcourses.xsd` - Definuje strukturu XML dokumentu s daty o kurzech.
  - `statistics.xsd` - Definuje strukturu XML dokumentu obsahující statistiky o obsahu dokumentu `allcourses_reduced.xsd`.
  - `acm_css.xsd` - Definuje strukturu XML dokumentu, jehož obsah je shodný s daty z ACM CCS.
- `style-xslt`: Tento adresář obsahuje XSL soubory se šablonami potřebné pro převod dat do jiných struktur. Dále obsahuje další soubory určené ke spuštění těchto XSLT šablon na operačním systému Microsoft Windows 7 a novější. Ke spuštění níže uvedených batch souborů je nutné umístit XML dokumenty s původními daty popsané v kapitole 2 do adresáře `data/original/`.
  - `allcourses_to_reduced.xsl` - Šablona pro převod dat z dokumentu `allcourses.xml` do `allcourses_reduced.xml`.
  - `all_merge_allcourses.xsl` - Tato šablona zařídí sloučení všech původních XML dokumentů s daty do jednoho dokumentu, viz kapitola 2.
  - `all_merge_allcourses_reduced.xsl` - Tato šablona zařídí sloučení všech redukovaných XML dokumentů s daty do jednoho dokumentu, viz kapitola 2.

  - `run_all_merge_allcourses_reduced.bat` - Batch soubor určený ke spuštění procesu sjednocení redukovaných dat. Výsledek je uložen do `data/reduced/allcourses_reduced.xml`.
  - `run_all_merge_allcourses.bat` - Batch soubor určený ke spuštění procesu sjednocení púvodních dat. Výsledek je uložen do `data/original/allcourses.xml`.
  - `run_allcourses_to_reduced.bat` - Batch soubor určený ke spuštění procesu transformace původních do redukovaných dat. Nevyžaduje provedení procesů pomocí výše uvedených batch souborů. Výsledek je uložen do `data/reduced/allcoursesPORADOVECISLO.xml`, kde místo XX je číslo shodné s původním souborem.
  - `saxon9he.jar` - Procesor XSLT.

- `zdrojove-kody`: Tento adresář obsahuje všechny zdrojové kódy rozdělené podle projektů (webová aplikace a vytvořené knihovny).
- `diplomova-prace`: Obsahuje všechny soubory nutné pro přeložení textu této diplomové práce.
- `videonavod`: V tomto adresáři je videonávod s ukázkou fungování aplikace.
- `stoplist`: Soubor se stop slovy je uložen v textovém formátu a umístěn v tomto adresáři.
- `ostatni`: Obsahuje textové soubory s doplňujícími informacemi pro nastavení webové aplikace k dalšímu vývoji.

A.2 Výpis výsledků hledání

Následuje výpis výsledků hledání realizovaných za pomoci analyzátoru `StandardAnalyzer`. Tento analyzátor byl použit pro proces indexace dat i pro následné vyhledávání.

1. **Bez rozšíření dotazu**

   Institution: *Arkansas Tech University*
   Title: *Computer networks*
   Department: *INFT*

   Study of the concepts involved in interconnecting computers. Introduction to **network** topologies, routing, protocols, and security. Survey of **network** operating systems.

2. **Rozšíření dotazu s WordNet**

   Institution: *Purdue University*
   Title: *Adv Network Admin*
   Department: *CIT*

   Credit Hours : 3.00. In this course, students will learn advanced concepts for installing, configuring and securing various types of **network** servers including enterprise, **web** and mail servers. The course also covers the documentation of **network** systems infrastructure and the testing of hardware and software **network** components. Typically offered Fall.

3. **Rozšíření dotazu s ACM CCS**

   Institution: *George Mason University*
   Title: *Network Security I*
   Department: *IT*

   Introduction to basic concepts in synchronous and asynchronous data communications, **network architectures**, protocols and current technology. Addresses basic **network** and information security issues and protection schemes with laboratory exercises designed to expose students to a variety of **network architectures**, security threats and risk mitigation strategies.

A.3 Doplňující údaje k databázi

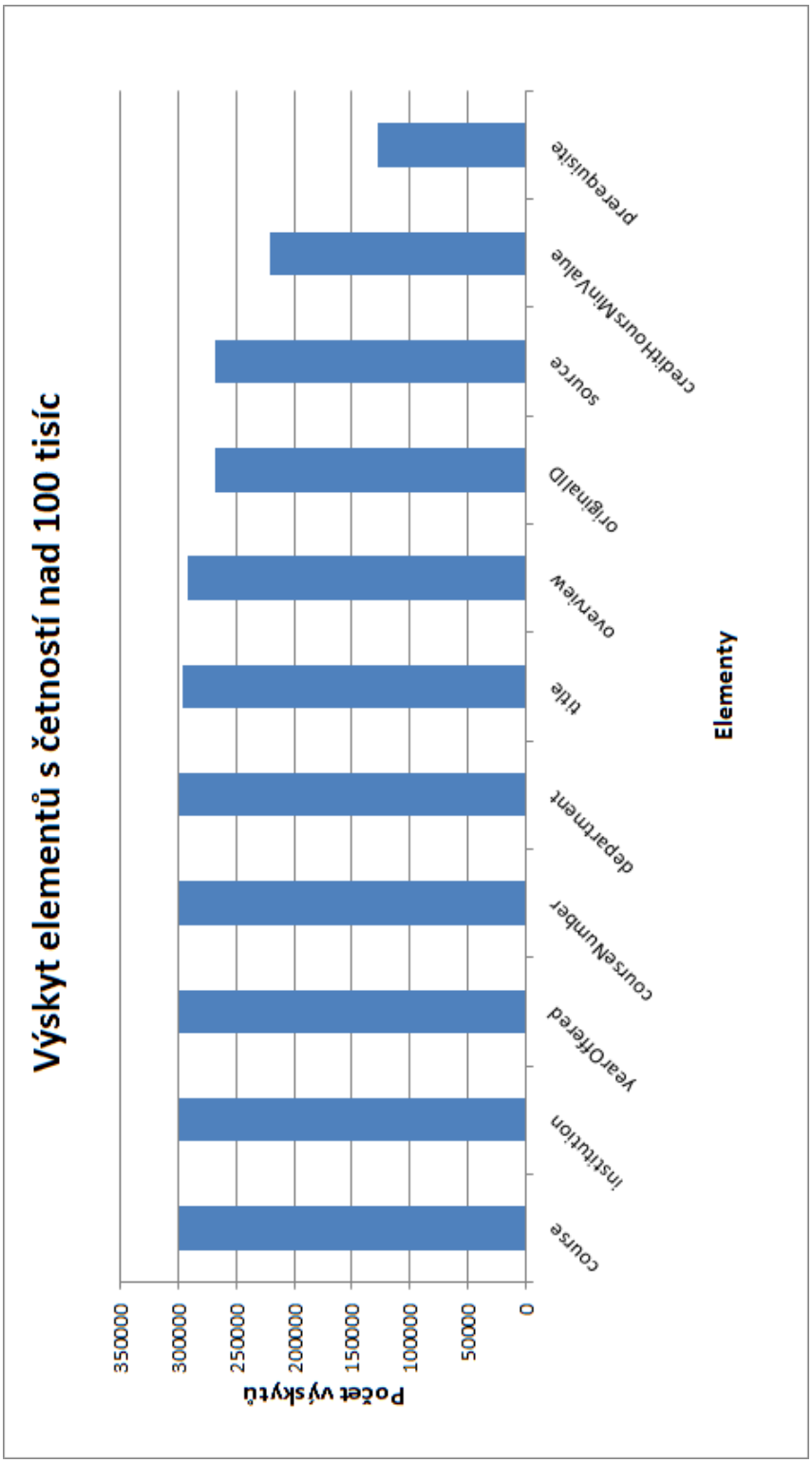
V následujících grafech a tabulkách jsou uvedeny všechny elementy společně s počtem jejich výskytů v databázi a krátkého popisu, které kurz může obsahovat. Nejprve jsou uvedena tabulky a potom grafy.

Element	Počet	Popis
yearOffered	299519	Nabízený rok kurzu.
institution	299519	Název univerzity.
department	299512	Zkratka katedry.
courseNumber	299512	Identifikace kurzu (i nečíselná).
title	296791	Název kurzu.
overview	292594	Popis kurzu (sylabus).
originalID	268061	Identifikace kurzu společně se zkratkou viz department.
source	267876	URL kurzu (nebyl nalezen platný odkaz).
creditHoursMinValue	220885	Min. kreditové ohodnocení kurzu.
prerequisite	127094	Prerekvizity.
creditHoursMaxValue	31623	Max. kreditové ohodnocení kurzu.
notes	28326	Obecný popis kurzu.
lectureHoursMinValue	26645	Min. hodinové ohodnocení přednášky.
labHoursMinValue	25655	Nabývá číselných hodnot.
summer	20334	Nabývá hodnot true a false.
fall	17304	Nabývá hodnot true a false.
levelDescription	17124	Určuje úroveň vzdělání (např. undergraduate).
spring	16918	Nabývá hodnot true a false.
gradingBasis	16519	V současné databázi se vykytuje hodnota regular.
name	14519	Shodný s elementem institution.
date	14519	Datum - význam neurčen.
xListings	12758	Textový popis - význam neurčen.

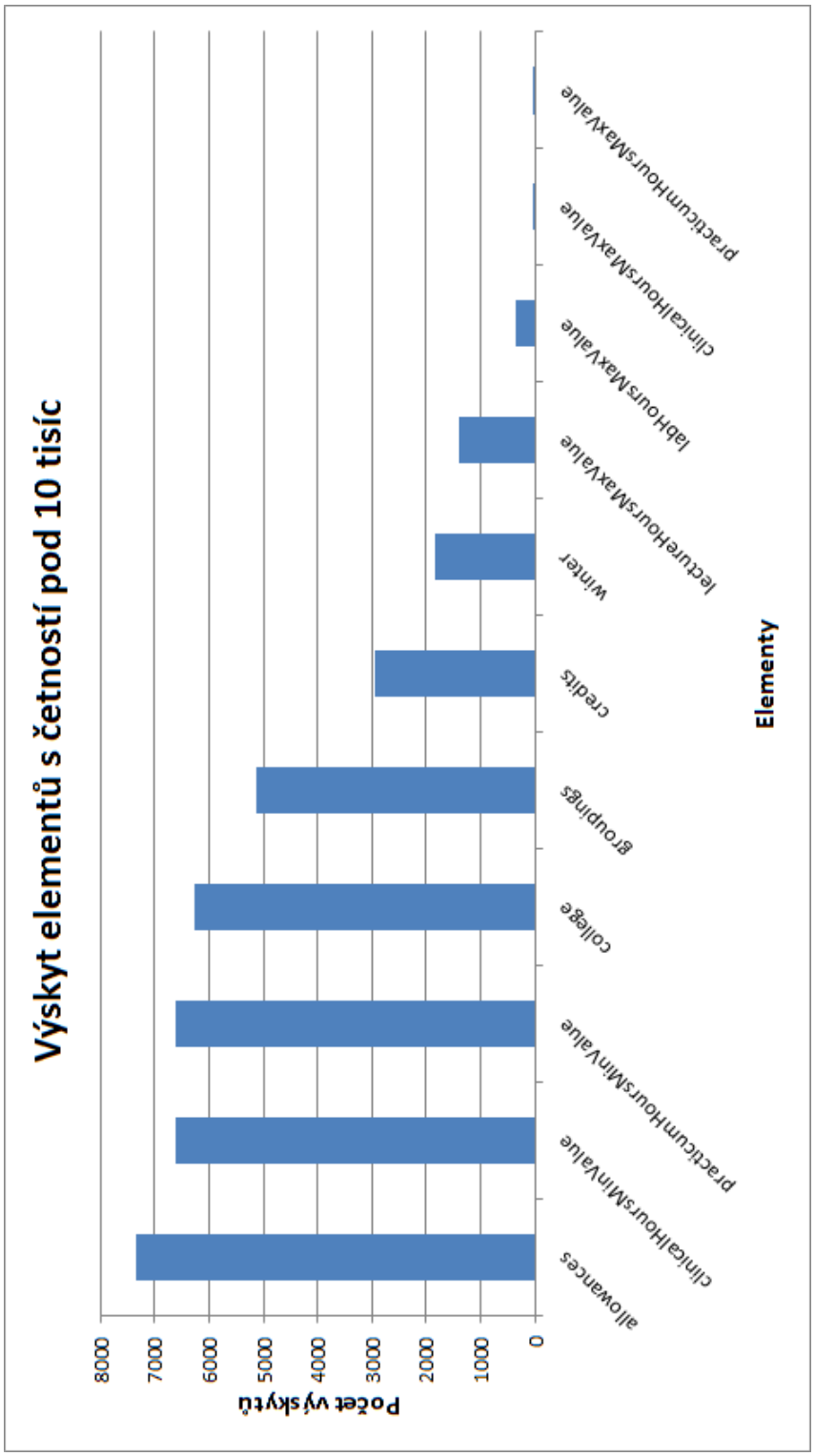
Tabulka A.2: Seznam elementů s počtem výskytů. Část první.

Element	Počet	Popis
corequisite	11561	Blíže nespecifikováno.
allowances	7345	Popis zda kurz může být opakován.
clinicalHoursMinValue	6612	Nabývá číselných hodnot.
practicumHoursMinValue	6610	Nabývá číselných hodnot.
college	6262	Blíže nespecifikováno.
groupings	5113	Blíže specifikuje zamìření kurzu.
credits	2937	Kreditové ohodnocení kurzu.
winter	1846	Nabývá hodnot true a false.
lectureHoursMaxValue	1394	Max. hodinové ohodnocení přednášky.
labHoursMaxValue	339	Nabývá číselných hodnot.
clinicalHoursMaxValue	4	Nabývá číselných hodnot.
practicumHoursMaxValue	2	Nabývá číselných hodnot.

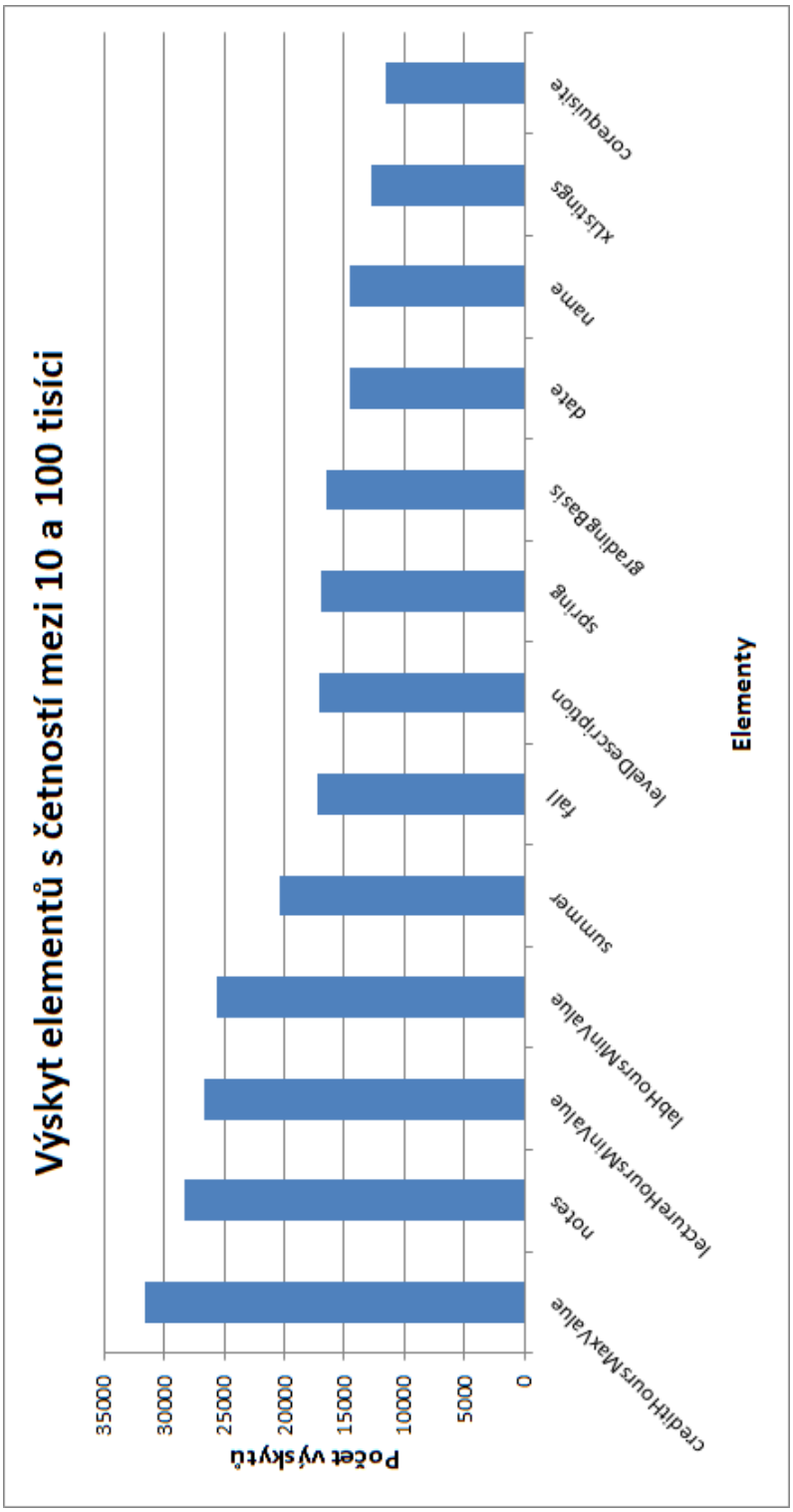
Tabulka A.4: Seznam elementů s počtem výskytů. Část druhá.



Obrázek A.1: Graf elementů s četností nad sto tisíc, viz tabulka A.2



Obrázek A.2: Graf elementů s četností pod deset tisíc, viz tabulka A.4



Obrázek A.3: Graf elementů s četností mezi deseti a sto tisíci, viz tabulky A.2 a A.4