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

Diplomová práce

Systém porovnání studijních programů českých univerzit

Plzeň 2017 Martin Hospaska

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10. května 2017

Martin Hospaska

Abstract

A system for comparing courses of selected universities

The goal of this diploma thesis is to create a web crawler and corresponding user interface, which presents the data obtained from the crawler. The created project also consists of a data storage used for preserving the retrieved data from the crawler. Target users of this platform are students. The aforementioned system should help them with comparing units taught at different universities. For this, the platform needs to be deployed and available on the public network.

This paper describes how the units are collected using the implemented crawler and how the collected data are interpreted. For this, courses at selected universities in the Czech Republic are used as testing data. In more details, the process of downloading the information from universities pages and storing it in the data storage is explained. Finally, the way how the collected data are compared is described.

In the scope of this thesis, the structure of pages of selected universities is described and the steps how to create a custom web crawler to collect data from these pages are given as well. The reader can also learn about different ways how the data can be exported. The steps how to create a custom user interface presenting the collected data are also given in the thesis.

Abstrakt

Systém pro porovnávání předmětů vybraných univerzit

Výstupem této práce je aplikace webového crawleru a vlastní webové stránky, které slouží pro zobrazení dat získaných pomocí crawleru. Součástí práce je rovněž vybrané datové úložiště sloužící pro uchovávání a vyhledávání v uložených datech. Po nasazení této práce na veřejně přístupný server je tak možné zájemcům o studium na vysoké škole nabídnout unikátní systém pro porovnávání předmětů vybraných univerzit.

Text práce popisuje způsob vytvoření a následné využití kolekce předmětů vyučovaných na vybraných vysokých školách v ČR. Zaobírá se jak samotným stahováním informací z webů univerzit, tak jejich uložením v databázi a následným porovnáváním předmětů pomocí vytvořených webových stránek.

Čtenář se v textu práce dozví, jakou strukturu mají webové stránky vybraných univerzit a jakým způsobem lze sestavit webový crawler, který dokáže z těchto stránek získávat data. Dozví se také o možnostech, jak stažená data uložit a následně s nimi pracovat. Pokud by si čtenář přál vytvořit webové stránky zobrazující stažená data, i o tomto se v textu dočte.

Obsah

1 Úvod

Cílem této práce je vytvoření webových stránek, přes které bude možné vyhledávat v kolekci informatických předmětů vyučovaných na vybraných českých vysokých školách. Tyto stránky tak budou sloužit převážně žákům středních škol ve vyšších ročnících studia při rozhodování, která univerzita svou skladbou předmětů nejlépe vyhovuje jejich představám, jakým směrem se bude jejich následující studium ubírat. Zájemce si na vytvořeném webu, vyhledáváním pomocí klíčových slov, zobrazí seznam předmětů, včetně jejich detailů. Ty bude moci následně procházet a označit si ty předměty, které ho zajímají. Ve finále pak získá jakési doporučení či alespoň představu, na kterou univerzitu by mohl podat přihlášku ke studiu. Toto doporučení bude však založeno pouze na základě vybraných předmětů, tedy jejich popisech a případné kreditové dotaci, nezohledňující faktory jako „prestiž“ univerzity, finanční dostupnost studia atd.

V textu práce jsou nejprve čtenáři krátce představeny podobné systémy pro agregaci dat z více zdrojů, následované kapitolou 3 se stručným popisem vybraných univerzit. Dále mu jsou vysvětleny teoretické pojmy týkající se této problematiky (kapitoly 4, 5 a 6). Teoretická část je pak zakončena kapitolou 7, popisující návrh vytvářeného systému.

Prvním krokem této práce je tedy utvoření kolekce informací o předmětech z vybraných univerzit. Informace budou stahovány z webových stránek jednotlivých univerzit. Toho bude dosaženo naprogramování vlastního webového crawleru, který ze stránek univerzit stáhne (vždy pro jednotlivé katedry) detailnější informace o vyučovaných předmětech. Touto částí se zaobírá sekce 8.1 věnující se tvorbě crawleru. Ten získá požadované informace a uloží je do vybraného datového úložiště (databáze). Z něj lze poté získávat informace dotazováním přes vytvořené webové stránky, resp. přes vytvořený server.

Druhým krokem k dosažení cíle je pak konfigurace a nasazení úložiště a serverové aplikace, přes kterou bude probíhat komunikace mezi uživatelem a úložištěm. Vzhledem k tomu, že lze očekávat poměrně velké množství předmětů na každé z univerzit, bude vhodné vybrat takové úložiště, které dokáže v reálném čase co nejrychleji vyhledávat fulltextově. Tedy co nejrychleji procházet velké množství dokumentů a hledat v nich nejlepší shodu se zadaným výrazem. Nastavením úložiště, vytvořením serveru a klientské aplikace se zabývají sekce 8.2, 8.3, respektive 8.4.

2 Rešerše agregačních systémů

V této kapitole jsou stučně popsány možnosti, jaký způsobem mohou pracovat webové stránky, zabývající se slučováním informací z několika různých zdrojů. Takovéto weby většinou nazýváme *agregátory* informací. Viz zdroje [21] a [26].

Tyto agregátory mohou obsahy svých stránek plnit z vlastních databází či sloužit pouze jako „zobrazovače“ informací ze stránek druhých stran. Zmíněné „zobrazovače“ se taktéž nazývají jako *RSS* (Rich Site Summary) *agregátory*. Pokud agregátory využívají vlastní databáze, mají následující možnosti plnění databáze daty: buď ji mohou plnit ručně, což je obvykle náročné na lidské a tedy i finanční zdroje, nebo ji mohou plnit automatizovaně. Automatizované plnění poté může probíhat způsobem *dolování dat* z webů druhých stran, tzv. *crawlingem*, či domluvou mezi agregačním webem a jeho „zákazníky“. Ti mu sami z vlastní iniciativy poskytují informace v jednotné podobě. Ta je předem domluvena a pokud chtějí mít „zákazníci“ zahrnuté své vlastní nabídky v rámci agregátoru, musí je sami poskytnout.

Předmětem zájmů těchto agregátorů může být dnes prakticky cokoliv. Nejvíce zastoupeny jsou zde weby srovnávající ceny, nabízející služby či různé slevové portály. Na internetu lze nalézt spoustu různých portálů na nabízení slev, portály porovnávající ceny zboží, ceny hotelů, letenek atd. Dalšími mohou být například agregátory pro zobrazení pracovních nabídek či v neposlední řadě například seznamky.

Pokud jde o vyhledání informací, jakým způsobem pracují konkrétní agregátory, jsou dostupné informace bohužel velmi málo specifické. Pouze u několika málo agregátorů se podařilo zjistit, jakým způsobem svá data získávají.

2.1 Systémy vyhledávání cen zboží

V dnešní době je v ČR provozováno několik známých systémů, které fungují na principu sběru informací z jiných webů a tyto informace následně využívají pro svou funkčnost na vlastních webových stránkách. Jedná se o weby, na kterých si lze srovnat ceny požadovaného zboží, které lze zakoupit v interne-

tových obchodech. Informace byly čerpány ze zdrojů [30] a [12].

Mezi tyto systémy patří například `www.heureka.cz`, `www.zbozi.cz` nebo `www.hledejceny.cz`. Všechny tyto zmíněné systémy fungují na velmi podobném principu. Od partnerských webů (tedy konkrétních internetových obchodů, které chtějí mít své zboží „vystavené“ v těchto systémech) vyžadují pravidelný tzv. *XML* (Extensible Markup Language) *feed*, což je soubor ve formátu XML popisující dané produkty partnerského internetového obchodu. Tento XML soubor obsahuje mnoho informací, nejen název a cenu produktu. Soubor je zpravidla stahován (aktualizován) jednou denně a následně zpracováván roboty jednotlivých vyhledávačů zboží.

Pro systémy vyhledávání zboží je tento způsob velmi výhodný, protože nejdůležitější část své práce tak přenáší na partnerské obchody, které – pokud si přejí být zobrazeny – musí dodávat aktuální a validní XML soubory svého zboží. Vyhledávače pak již velmi jednoduše tyto XML soubory *rosparsují* a zpracují dle potřeby.

2.2 Slevové portály

Dalšími dnes velmi oblíbenými portály jsou ty, které se zaměřují na nabízení slev, ať již přímo na zboží či různé zážitkové poukazy. Nabízejí se na nich již hotové produkty, služby, zájezdy či zážitkové akce.

K takovýmto portálům lze přiřadit tuzemské nejznámější weby, jakými jsou například `www.slevomat.cz`, `www.slevin.cz` nebo `www.skrz.cz`. Zmíněné systémy fungují převážně opět na tzv. *affiliate přístupu*, což je pouze jiné pojmenování pro dříve jmenovaný způsob získávání dat do svého systému. To znamená, že jsou nabídky pravidelně aktualizovány o informace dodané zákazníky, kteří mají zájem na tom, mít své nabídky zobrazeny v agregátoru.

Opět jsou zde při získávání informací pro agregátory zastoupeny možnosti jak vlastního crawlingu webů druhých stran, tak domluvy obou stran na vzájemné spolupráci a tvorbě XML feedu.

3 Vybrané univerzity

Tato kapitola se zabývá popisem vybraných univerzit. Stručně objasní, které univerzity a informace jsou zájmem této práce a ve finále shrne překážky při stahování požadovaných informací z webů univerzit.

Práce se aktuálně zaměřuje výhradně na předměty vyučované na českých vysokých školách. Pokud webové stránky se seznamy předmětů na univerzitách umožňovaly zvolit jazyk výuky, byly vybrány předměty vyučováné pouze v českém jazyce. Vyhledávání v databázi je proto také nastaveno na český jazyk. Případnému rozšíření o anglické předměty se věnuje sekce 9.2 na straně 80.

K integraci do systému bylo vybráno 10 českých univerzit, pod které spadají fakulty zabývající se výukou informatických předmětů, či ty univerzity, na jejichž fakultách se informatické předměty vyučují z větší části všech předmětů. Níže vyjmenované univerzity byly zvoleny na základě předchozích prací zabývajících se touto oblastí, které, stejně jako tuto práci, vedl Prof. Ing. Karel Ježek CSc. Nepovažoval jsem za nutné tento seznam upravovat, pro demonstraci funkčnosti práce je těchto 10 univerzit dostačující.

V budoucnu, či pokud by měla některá z dalších univerzit zájem přidat se k tomuto projektu, není problém tuto práci rozšířit o crawling dat z dalších univerzit. Stejně tak by, v případě zájmu o tuto práci, mohly do projektu přibýt univerzity ze zahraničí. Toho lze docílit jednoduchým rozšířením naprogramovaného crawleru a menšími úpravami webových stránek vytvořených v rámci tohoto projektu – blíže bude toto rozšíření popsáno ve zmíněné sekci 9.2 v závěru práce.

Práce se nyní, z časových důvodů, zaměřuje pouze na předměty informatické, respektive ty, které jsou vyučovány na vybraných informatických fakultách jednotlivých univerzit. V budoucnu opět není problém rozšířit systém i o předměty z jiných fakult.

3.1 Seznam univerzit

Tento seznam obsahuje stručný popis vybraných univerzit a jejich fakult, včetně odkazu, který slouží jako startovní bod crawleru pro každou z uve-

dených fakult. Jak bylo zmíněno v textu výše, práce se aktuálně zaměřuje pouze na fakulty vybraných univerzit, které vyučují převážně informatické předměty.

3.1.1 České vysoké učení technické v Praze

- Fakulta informačních technologií (FIT)
- Na adrese `http://bk.fit.cvut.cz/cz/prehled.html`, spadající přímo pod zmíněnou fakultu, nalezneme rozcestník, ve kterém lze různými způsoby procházet studijní plány a předměty zde vyučované. Pomocí jednoduchého textového menu v horní části webu se lze postupně dostat až na stránku se seznamem jednotlivých kateder. Z té lze přejít na část, v níž se nachází seznam všech předmětů vyučovaných na této katedře. Předmety jsou rozděleny do jednotlivých tabulek, dle fakult, pod které spadají. Nachází se zde jak předmety vyučované v češtině, tak v angličtině. V případě této univerzity není možno je vyfiltrovat pouze pro jeden z jazyků, proto jsou stahovány jak česky, tak anglicky vyučované předměty. Z tohoto seznamu lze poté čerpat přímo odkazy na konkrétní předměty, které obsahují detailnější informace.

3.1.2 Masarykova univerzita Brno

- Fakulta informatiky (FI)
- Informační systém univerzity se nachází na adrese `https://is.muni.cz/`. Z tohoto bodu se lze jednoduše dostat na katalog pro výběr vyučovaných předmětů. Dále je nutno z nabízeného výběru zaškrtnout odpovídající fakultu a poté zvolit všechny obory z fakulty. Tím dostaneme seznam všech předmětů.

3.1.3 Technická univerzita Liberec

- Fakulta mechatroniky, informatiky a mezioborových studií (FM)
- Tato univerzita využívá jednotného systému ECTS (European Credit Transfer and Accumulation System, volně přeloženo jako *kreditní systém*), stejně jako ZČU (3.1.10 na straně 8) , kde je tento systém

blíže popsán. Počáteční adresa je následující: `http://ects.tul.cz/search?lang=cs`.

3.1.4 Technická univerzita Ostrava

- Fakulta elektrotechniky a informatiky (FEI)
- Z úvodního webu `https://www.vsb.cz/cs` se lze pomocí menu jednoduše proklikat přes položky „Uchazeči“ a dále „Studijní programy“ ke katalogu, ve kterém lze filtrovat jednotlivé fakulty a získat tak seznam předmětů. Požadovaným výběrem dostáváme vyfiltrované studijní plány, po jejichž výběru se dostaneme na seznam jednotlivých předmětů daného plánu. Seskupením všech těchto plánů dostaneme úplný seznam všech vyučovaných předmětů na dané fakultě.

3.1.5 Univerzita Karlova

- Matematicko-fyzikální fakulta (MFF – většina informatických předmětů je zde vyučována pod touto fakultou)
- Na adrese `https://is.cuni.cz/studium/index.php` je možné najít informační systém Univerzity Karlovy. Zde nalezneme odkaz na formulář pro vyhledávání předmětů. Po načtení formuláře zvolíme požadovanou fakultu. Dále je nutno taktéž navolit kategorizaci předmětu, v tomto případě „Informatika“. Tím jsou výsledky omezeny na informatické předměty, jejichž seznam získáme odesláním formuláře.

3.1.6 Univerzita Pardubice

- Fakulta elektrotechniky a informatiky (FEI)
- Univerzita využívá jednotného systému ECTS, stejně jako ZČU (3.1.10 na straně 8) , kde je tento systém blíže popsán. Počáteční adresa je následující: `http://ects.upce.cz/fakulty/FEI/KIT?lang=cs`.

3.1.7 Univerzita Tomáše Bati ve Zlíně

- Fakulta aplikované informatiky (FAI)
- Univerzita využívá jednotného systému ECTS, stejně jako ZČU (3.1.10 na straně 8) , kde je tento systém blíže popsán. Počáteční adresa je následující: `http://ects.utb.cz/?lang=cs`.

3.1.8 Vysoká škola ekonomická v Praze

- Fakulta informatiky a statistiky (FIS)
- Z hlavních stránek univerzity na adrese `https://www.vse.cz/` se přes odkaz „Integrovaný studijní informační systém“ a dále přes „Katalog předmětů“ dostaneme do katalogu předmětů. Zde nejprve volíme rozšířené vyhledávání. Poté si opět pomocí formuláře zvolíme rok, semestr (nutno vybrat oba semestry daného roku) a fakultu, pro kterou chceme vyhledat předměty pod ní spadající. Odesláním tradičně získáme seznam předmětů pro konkrétní katedru.

3.1.9 Vysoké učení technické v Brně

- Fakulta informačních technologií (FIT)
- Z adresy `https://www.vutbr.cz` se pomocí menu, přes volby „Studium“ a následně „Předměty“ dostáváme k formuláři, ve kterém vybereme požadovanou fakultu. Odesláním formuláře poté dostaneme seznam předmětů. Zvláštnost oproti ostatním univerzitám je na tomto webu v tom, že seznam předmětů je zde rozdělen na několik „podstránek“. Pro každou z nich je uvedeno maximálně 15 předmětů. Při procházení stránek je tak nutno mezi těmito částmi přepínat, což byla hlavní odlišnost tohoto webu při tvorbě crawleru. Celkový seznam předmětů dostaneme opět jednoduše sumou všech předmětů ze všech „podstránek“ webu.

3.1.10 Západočeská univerzita

- Fakulta aplikovaných věd (FAV)
- Západočeská univerzita využívá, jako několik výše zmínených univerzit, jednotného systému ECTS pro prohledávání studijních plánů a katalogu předmětů. V levé části úvodní stránky se nachází seznam jednotlivých fakult. Po zvolení některé z nich nalezneme všechny studijní plány dané fakulty, přes které se dále dostaneme na studijní obory. V nich se pak nachází samotné odkazy na jednotlivé předměty. Tuto stromovou strukturu je tedy nutné projít celou, pro každou položku, abychom získali výslednou sumu vyučovancýh předmětů. Pro ZČU je počáteční adresa tohoto informačního systému `http://ects.zcu.cz/browser/FAV?lang=cs`.

3.2 Problém struktury webů univerzit

Jak bylo zmíněno v předchozí kapitole (kapitola 2 na straně 2), mají některé vyhledávače zboží práci velmi usnadněnou tím, že od partnerů získávají informace v pevně definované struktuře, kterou si dle svých potřeb určí předem. Partneři pak již sami zasílají data, která stačí následně parsovat a zobrazovat.

V případě seznamu předmětů vyučovaných na českých vysokých školách bohužel žádný podobný jednotný systém nefunguje a téměř každá univerzita si tak seznam předmětů udržuje svým vlastním způsobem. Jen několik škol využívá společného systému ECTS, avšak z vybraných 10 univerzit jsou to pouze 4. To znamená nejen různou hierarchickou strukturu uložení informací na webových stránkách, ale také různě pojmenované (či v horším případě úplně chybějící) informace o vyučovaných předmětech samotných. U některých univerzit například došlo k tomu, že většina předmětů obsahuje prázdné pole pro anotace k předmětu, či tuto informaci neobsahuje vůbec. Tento problém je patrný především u univerzity Vysoké učení technické v Brně. Detaily předmětů zde obsahují velmi malé množství informací (požadované kolonky jsou zde uvedeny, ale nejsou vyplněny) a potřebné informace se jinde na webových stránkách univerzity nepodařilo dohledat.

V takovém případě je při crawlingu místo chybějících informací vyplněno pouze slovo „neuvedeno“. To ovšem při následném vyhledávání v seznamu všech předmětů penalizuje jak dané univerzity, jejichž předměty se díky

tomu ve vyhledávání pravděpodobně nikdy nezobrazí, tak hlavně uživatele, kteří jsou zbytečně ochuzeni o další předměty, které by je potenciálně mohly zajímat.

Prvním hlavním problémem této práce je vytvoření samotného crawleru, který bude muset být naprogramován „na míru“ pro každý jednotlivý web univerzit. Je totiž nutné zajistit, aby byl schopný procházet větší množství různě strukturovaných webů vybraných univerzit. Získané informace budou poté ukládány do úložiště jednotným způsobem, který z různě pojmenovaných, avšak ve výsledku stejných, informací utvoří stejnorodý celek.

Ideálním stavem by bylo dosáhnout něčeho podobného, co se podařilo zmíněným systémům srovnávání cen – tedy aby se jednotlivé univerzity samy angažovaly v tomto projektu a poskytly vždy aktuální, platné a stejně strukturované seznamy svých předmětů na požádání. V lepším případě by toto mohlo být navíc automatizované, kdy by byly seznamy získávány v pravidelných časových intervalech v podobě zmíněných XML feedů. Dosažení tohoto stavu však není cílem této práce a pravděpodobně by si toto rovněž vyžádalo i více času k realizaci celé myšlenky.

4 Web crawling

Tato kapitola blíže objasní pojmy jako je *web crawling*, *crawler* a jeho různé způsoby či možnosti využití. V případě crawleru se jedná o první velmi důležitou část této práce, bez níž by bylo vytvoření kolekce předmětů prakticky nemožné. Text k této kapitole byl čerpán na základě zdrojů [15], [29] a [4].

4.1 Co je to crawler

Crawler je typ programu, který zvoleným způsobem prochází vybrané webové stránky, z nichž následně stahuje jak požadované informace (text), tak různá metadata o těchto informacích, případně i multimediální a datové soubory. Zároveň by si také měl při procházení webu ukládat všechny odkazy, které se na stránce vyskytují, aby bylo možné navštívit web „do větší hloubky“. To je důležité především také z toho důvodu, aby nedošlo k „zacyklení“ běhu programu, kdy by crawler stále dokola procházel stejné, již navštívené, odkazy. Zde však záleží na tom, k jakému účelu bude crawler sloužit. Pokud se bude jednat např. o pravidelné opakované procházení pouze jediného URL (Uniform Resource Locator – jednotný identifikátor zdrojů v internetu) bez dalšího přecházení mezi odkazy, není tak samozřejmě nutné si uchovávat všechny odkazy z tohoto URL.

Hlavním cílem crawleru je tedy připojit se na zadanou adresu URL, stáhnout obsah stránky a následně tuto stránku požadovaným způsobem zpracovávat – ať již na ní vyhledávat požadované informace, či nacházet odkazy na další stránky, které by tyto informace mohly obsahovat.

Taktéž si crawler musí umět poradit s různými návratovými hodnotami stránek, tedy například v případě, že URL odkazující na další stránku je neplatné, či hledaná stránka neexistuje nebo je nepřístupná. Dalším problémem může být například odkaz, který obsahuje několik parametrů, které mohou mít různé pořadí – crawler tak vidí dva různé odkazy, avšak může se jednat o ten samý odkaz, pouze zapsaný jiným způsobem. Tento problém byl řešen i v rámci této práce – při crawlingu bylo získáno několik odkazů pro stejný předmět. Řešením je zde uchovávání seznamu zkratek předmětů a v případě nalezení již získané zkratky přeskočit daný odkaz.

4.2 Typy crawlerů

Osobně jsem webové crawlery rozdělil do několika kategorií, podle toho, jakým způsobem je možné je získat, respektive využívat. V rámci této práce jsem se setkal právě s těmito níže zmíněnými možnostmi a vyzkoušel si všechny níže vypsané způsoby.

4.2.1 Nástroje s vlastním grafickým rozhraním

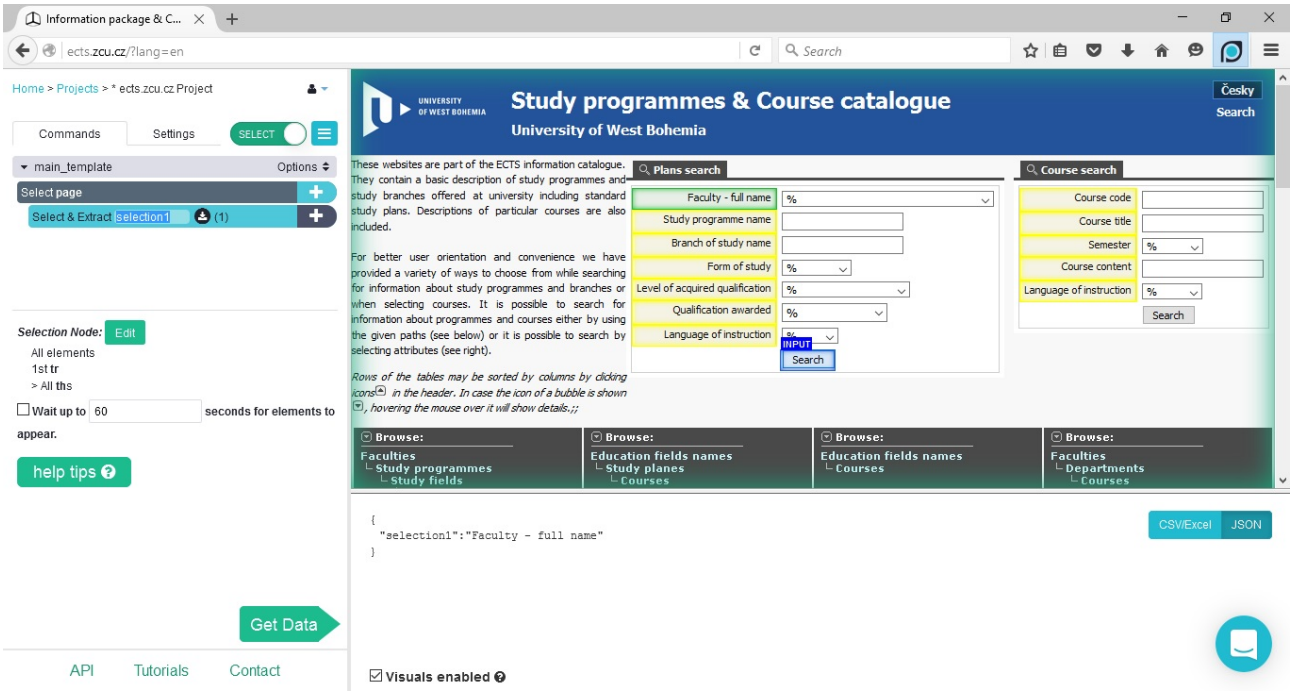
V tomto případě se jedná o komplexní řešení, kdy uživatel získá program s vlastním grafickým rozhraním, které je z popsaných řešení nejpřívětivější na ovládání. V rámci této práce byl k účelu procházení webu vyzkoušen (na základě [24]) program *ParseHub* (dostupný ke stažení na adrese `https://www.parsehub.com/`).

Po bezplatném stažení, instalaci a spuštění je uživatel vyzván k založení projektu. Poté už je přímo v uživatelském rozhraní programu zobrazena zadaná webová stránka, kterou chce uživatel crawlerem procházet. Na ní si pouhým poklepáním tlačítka myši označí, která data chce ze stránky stahovat a jakým způsobem budou ukládána do výstupního souboru, který je poté výsledkem crawlingu. Program se sám snaží odhadnout způsob, jak jsou vybraná data na webu uspořádána a označuje tak vše, co předpokládá, že by chtěl uživatel stahovat. Výstupním formátem crawlingu tohoto programu je soubor ve formátu *JSON* (JavaScript Object Notation) či soubor s příponou *csv* (Comma-separated values).

Hlavním problémem tohoto přístupu je však zmíněný odhad, který ne vždy dokáže správně určit způsob zobrazení dat na webu a například v případě pokusu o stažení seznamu předmětů z webu ČVUT naprosto selhal. Možnou příčinou je však pouze neznalost tohoto konkrétního programu. Dalším nedostatkem je také to, že se jedná o komerční program (byť se základem zdarma), ve kterém jde založit zdarma pouze 5 projektů (kdy jeden projekt odpovídá jednomu procházenému webu), což při počtu minimálně deseti vybraných univerzit bohužel nestačí.

Představu, jak zmíněný program vypadá, ukáže obrázek 4.1 níže. V jeho levé části je zobrazena hierarchická struktura aktuálně vyselektovaných informací. Prostřední část je věnována webu, ze kterého chceme získat informace (zde ECTS katalog Západočeské univerzity). V této části lze poklepáním myší

vybírat jednotlivé elementy webu, které budou stahovány. Dolní část obrazovky pak ukazuje, jakým způsobem bude vypadat výstupní soubor (možno volit mezi JSON a csv dokumentem).



Obrázek 4.1: Ukázka programu ParseHub.

4.2.2 Nástroje bez grafického rozhraní

Toto řešení bylo zkoušeno jako druhé v pořadí. V tomto případě se jedná o nástroje, jejichž nastavení se provádí úpravou textových konfiguračních souborů nacházejících se v adresáři rozbaleného nástroje. Spuštění nástrojů se následně provádí přes systémovou příkazovou řádku. V rámci této práce byla vyzkoušena dvě různá řešení. Prvním z nich je nástroj Apache Nutch (verze 2.3.1., dostupný ke stažení z adresy `http://nutch.apache.org/`). Druhým pak Norconex HTTP Collector (verze 2.6.0., který lze získat na `https://www.norconex.com/`). Obě tato řešení jsou dostupná k použití zdarma.

ApacheNutch

První zmíněný nástroj je na internetových fórech hojně doporučovaný webový crawler, jehož hlavní výhodou je jednoduchost, s jakou ho lze propojit s jinými užitečnými nástroji (např. s databází a následným vyhledávacím enginem, viz [23]). Problémem je zde však přílišný rozsah tohoto projektu (po rozbalení staženého archivu zabírají na disku počítače data nástroje přibližne 750 MB) a neintuitivní ovládání, respektive nastavení celého procesu stahování stránek. To se mi ani po mnoha hodinách zkoušení nepodařilo nastavit požadovaným způsobem. Problém byl však opět neznalost tohoto nástroje, dokumentace a návody jsou na internetu snadno dohledatelné.

Norconex HTTP Collector

Druhé jmenované řešení, od společnosti Norconex, je řešení o mnoho kompaktnější (celý nástroj zabírá na disku počítače přibližně 100 MB), ne však méně schopné. Ovládání tohoto nástroje mi osobně přišlo snazší a intuitivnější, stejně tak jako nastavení všeho potřebného pro crawling stránek. Zde jsem však opět narazil na neznalost možností nástroje – i přes to, že šlo velmi svižně komunikovat přímo s tvůrci programu (komunikace přes projekt nástroje, který je založený a vyvíjený na úložišti *GitHub*, na adrese `https://github.com/Norconex/collector-http`), zde chyběly podrobnější návody pro začátečníky. Projekt by si určitě zasloužil pár jednoduchých návodů, jak celý systém nastavit, aby stahoval pouze požadované informace z vybraných stránek.

Po několika dnech zkoušení tohoto nástroje jsem se dobral k výsledku, kdy se mi podařilo ze „startovního“ URL získal odkazy na další navazující stránky a tyto následně procházet. Výsledkem procházení však byl pouhý *plain text* obsahující všechny texty z procházených stránek, v nestrukturované podobě. Takovéto výsledky by bylo nutné ještě dále zpracovávat vlastním programem, což by ve výsledku práci při stahování informací opět příliš neulehčilo. Přesnější texty by se pravděpodobně daly získat správým nastavením regulárních výrazů pro crawling stránek, stejně by však bylo nutné vytvořit tyto výrazy pro každou z univerzit zvlášť. Taková práce by byla pravděpodobně srovnatelná s napsáním vlastního crawleru, kde má programátor „vše pod kontrolou“ a stahuje požadované informace přímo. Proto bylo po pár dnech zanevřeno i na toto řešení.

4.2.3 Vlastnoručně napsaný crawler

Nakonec se tak v rámci práce využívá právě tohoto řešení, tedy vlastnoručně napsaného crawleru. Pro člověka, který nemá s výše popsanými prostředky pro crawling žádné zkušenosti, je toho řešení nakonec pravděpodobně časově nejvýhodnější (tedy alespoň v případě, že požaduje procházení pouze menšího počtu webů – zde 10 různých webů univerzit). Výše zmíněné nástroje jsou totiž určeny spíše pro stahování informací ze stránek v množství řádu statisíců až milionů. Při psaní vlastního crawleru je taktéž výhodou maximální kontrola nad tím, co se bude odkud stahovat. Nevýhodou může být naopak nutnost znalosti jazyka *HTML* (HyperText Markup Language) a přesné struktury stránek (více o struktuře stránek vybraných univerzit viz kapitola A na straně 90), které bude crawler zpracovávat. Nástrojům, či *knihovnám*, využívaných pro zhotovení vlastních crawlerů, se říká taktéž *HTML parsery*.

Po porovnání několika nabízených řešení na webu wikipedia.org[1] zde vychází jako jedna z nejvíce udržovaných knihoven, která zároveň nabízí nejlepší možnosti, knihovna *Jsoup*. Proto je v tomto projektu využito právě této knihovny (více informací na `https://jsoup.org/` a ve zdroji [27]), jejíž použití je velmi snadné a intuitivní.

Menší problém vyvstal u webů, kdy bylo nutné seznam předmětů získat odesláním formuláře až pomocí programovém kódu crawleru. Ve výsledku je řešení opět celkem jednoduché. V takovém případě stačí v libovolném internetovém prohlížeči otevřít vývojářskou konzoli (kterou mají dnešní nejpoužívanější prohlížeče již automaticky vestavěnou) a odeslat požadovaný formulář. V konzoli lze poté dohledat požadavek, který obsahuje odesílaná data. Z něj lze pak vyčíst jména a hodnoty parametrů, které formulář odesílá, a následně tyto informace použít v crawleru pro simulaci odeslání tohoto požadavku.

[1] Porovnání několika HTML parserů na adrese: `https://en.wikipedia.org/wiki/Comparison_of_HTML_parsers`.

5 Formát a uložení stažených informací

Tato kapitola popisuje možnosti a způsoby, jak lze získané informace pomocí crawleru uložit, aby v nich následně mohlo probíhat vyhledávání. Popsán bude jak formát, ve kterém jsou data stahována, tak možnosti jejich uložení.

5.1 Formát uložení dat

Nejprve je nutné zvolit si formát dat, ve kterém budou stažená data reprezentována a poté i uložena. Zde je možno volit z nepřeberného množství způsobů, jak lze výsledná data formátovat. Níže budou jmenovány dva dnes pravděpodobně nejznámější a nejpoužívanější formáty.

5.1.1 XML

Jako první vhodná možnost se nabízí formát *XML* [3]. Jde o hojně využívaný formát, jehož struktura je dána takzvanými *tagy*, mezi které jsou vkládány požadované informace. Tyto tagy jsou povinně párové, tedy každý tag má svou počáteční a ukončovací část. Tagy také mohou obsahovat různé parametry, které určují jejich další vlastnosti. Jako příklad XML lze uvést následující zápis, jak by mohl vypadat uložený předmět (pro ukázku uvedeny jen dva parametry):

```
<university>
    <subject>
        <name> PPA1 </name>
        <faculty> FAV </faculty>
    </subject>
    <subject>
        <name> INS </name>
        <faculty> FAV</faculty>
    </subject>
</university>
```

5.1.2 JSON

Druhou možností je formát *JSON* [28]. Jedná se o formát podobný zmíněnému XML, má však svá vlastní specifika. Data jsou v něm ukládána způsobem „Klíč : Hodnota“. Jeho předností je lepší čitelnost, respektive přehlednost, pro člověka, stejně jako výsledná velikost souborů – neuchovává „přebytečné“ párové (ukončovací) značky, jako zmíněné XML. Díky tomu může být stejná informace uložena s datovou úsporou. Příklad informace ve formátu JSON je následující (popisující stejný příklad jako výše uvedené XML):

```
{”university”: [
      ”subject”: [ {”name”:”PPA1”,”faculty”:”FAV”} ],
      ”subject”: [ {”name”:”INS”, ”faculty”:”FAV”} ]
  ] }
```

Z důvodů lepší čitelnosti a úspory dat byl tedy zvolen formát JSON. Zároveň je tento formát vyžadován výsledným zvoleným typem úložiště pro tuto práci, což je další plus pro jeho volbu. JSON je zároveň velmi dobře a snadno zpracovatelný ve webové části projektu, která využívá skriptovacího jazyka *AngularJS*.

5.2 Úložiště dat

Stažená data lze ukládat několika možnými způsoby. Nejjednodušší možností je jednotlivá data uložit jako prosté soubory na disk počítače. Tento způsob je však nepřehledný a hlavně velmi špatně spravovatelný.

Jelikož stažená data obsahují velké množství prostého textu, bude vhodné využití některé z dostupných dokumentově orientovaných databází. Ty jsou většinou již vybaveny prostředky pro ukládání a hlavně vyhledávání v takovýchto datech [34]. Pro tuto práci se tak jedná o velmi užitečný nástroj, který lze snadno využít, bez nutnosti mít hlubší znalosti o vyhledávacích algoritmech, indexování dokumentů a slov a podobných praktikách, které mívají tyto databáze v sobě zabudované. Tyto postupy jsou v nich již implementovány na velmi vysoké úrovni a vlastnoručně napsané algoritmy by velmi pravděpodobně nebyly ani rychlejší, ani efektivnější.

U dokumentově orientovaných databází se rovněž hovoří o takzvaných *bezschémových* (*schemaless* či *NoSQL*) databází [22], [33]. To znamená, že v databázi není potřeba vytvářet předem žádné schéma s pevně danou strukturou, jak mají uložené dokumenty vypadat. Každý záznam (dokument) tak v databázi může vystupovat v jiné podobě.

Množství dokumentů, které budou v rámci této práce ukládané, lze hrubým odhadem vypočítat následovně:
Aktuálně jsou data stahována z 10 univerzit, vždy na jeden aktuální studijní rok, za oba semestry. Odhadem vychází na jednu univerzitu (respektive její fakultu zabývající se informatickými předměty) průměrně 450 předmětů. Pokud bude tento projekt někdy rozšířen i o další fakulty, vynásobíme výsledek ještě číslem 9 (ZČU má právě 9 fakult).
Tím dostáváme výsledek: 10*450*9 = 40 500. Odhadem tedy vidíme, že ukládaných předmětů není mnoho, pokud si uvědomíme, že dnešní dokumentově orientované databáze jsou dimenzovány až na miliony takovýchto dokumentů.

Aktuálně je v databázi tohoto projektu staženo a uloženo téměř 4000 dokumentů obsahujících informace o předmětech. Data jsou stahována z 10 univerzit, u každé po jedné fakultě. To přibližně potvrzuje výše zmíněný odhad počtu dokumentů.

Na základě několika zdrojů byla nakonec blíže zkoumána tři možná řešení, podrobněji popsána v textu níže.

5.2.1 MongoDB

Mezi obecně nejznámější databázové řešení patří zajisté MongoDB (ke stažení na adrese `https://www.mongodb.com`). Jde o tzv. *NoSQL* databázi, která je orientovaná na ukládání dokumentů. Publikována je jako opensource, je tedy dostupná pro veškeré využití zcela zdarma. Jedná se však o poměrně komplexní řešení, zaměřené hlavně na uložení obrovského množství dokumentů (v řádech statisíců až milionů), což je pro účely této práce velmi předimenzované. Po instalaci zabírala tato databáze na disku počítače téměř 500 MB (prázdná DB, bez jakýchkoliv záznamů). To pět vypovídá o její zbytečné komplexnosti pro tuto práci, oproti dalším, níže zmíněným, řešením.

Databáze funguje jako klasický databázový server, spuštěný na určité adrese a portu. Na tuto adresu se lze poté připojit a komunikovat s da-

tabází. MongoDB využívá k ukládání data ve formátu JSON (respektive jeho binární podobou jménem BSON). K vyhledávání mezi dokumenty využívá speciálních příkazů. Příklad dotazu pro vyhledání konkrétních dokumentů vypadá následovně:

```
// Vyhledání předmětu s názvem „PPA1“
db.subject.find({name: PPA1})
```

Výsledkem dotazu je pak *kurzor*, pomocí kterého lze následně získané výsledky postupně procházet.

Vkládání a výběr dat z této databáze nebylo z časových důvodů vyzkoušeno. Přesněji řečeno, původně byla snaha propojit tuto databází s jedním z „automatizovaných“ crawlerů, avšak jak bylo zmíněno výše, nepodařilo se tento crawler uspokojivě zprovoznit, proto bylo nakonec od obou těchto řešení upuštěno.

5.2.2 RaptorDP

Jde o další z dokumentově orientovaných databází, která však oproti výše zmíněné MongoDB sází na svou jednoduchost a kompaktnost (archiv se soubory této databáze zabírá po stažení pouze několik stovek KB), což se pro tuto práci jeví jako příhodnější varianta. Jedná se o projekt vyvíjený rovněž jako opensource (`http://raptordb.codeplex.com/`), jež je stále aktivně udržován a aktualizován.

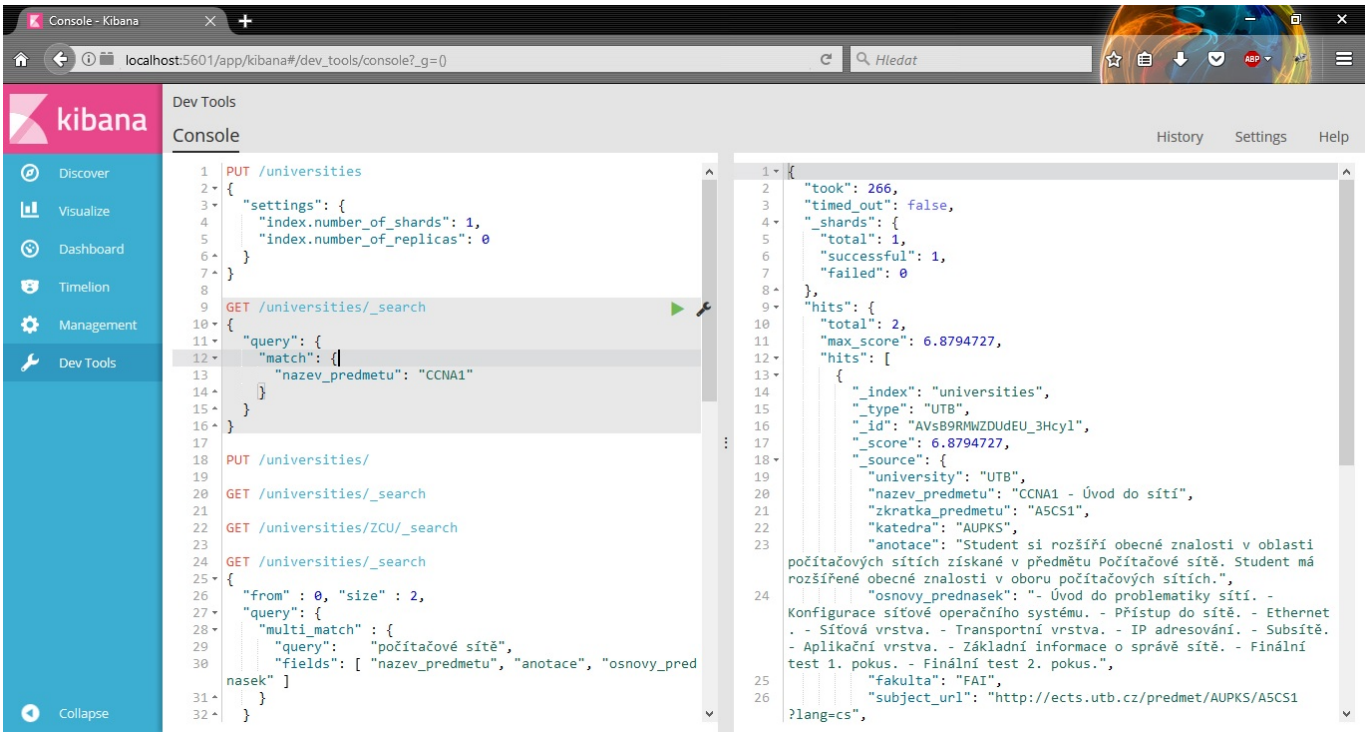
Jako formát souborů využívá opět dokumentů typu JSON. Stejně tak obsahuje vlastní fulltextový vyhledávací engine h00t. Tato databáze nebyla ve finále testována, je zde uvedena pouze jako jedna z několika dostupných možností, která se řadila mezi ty nejvhodnější.

5.2.3 ElasticSearch

Dalším z nabízených řešení je databáze ElasticSearch [32]. V případě tohoto nástroje se nemluví přímo o tom, že by se jednalo o druh databáze, pojmenován je spíše jako fulltextový vyhledávač. Což ovšem znamená, že „vnitřně“ jako dokumentově orientovaná databáze funguje – dokumenty musí být do

úložiště vloženy a zaindexovány, aby v nich následně mohlo probíhat vyhledávání [8].

Samotné vyhledávání se dle zdrojů řadí mezi jedno z nejrychlejších, které je (v programech dostupných zdarma) dnes možné získat, což je pro tuto práci velmi žádoucí. Tento nástroj je založen na technologii *Apache Lucene* a je tak šířen zdarma pod licencí *Apache*. Porovnání tohoto nástroje s *MongoDB* nabízí zdroj [11]. Jako komunikační rozhraní s tímto nástrojem slouží *REST API* (Representational State Transfer Application Programming Interface). Dokumenty jsou pak ukládány ve formátu JSON. Není tedy vůbec obtížné do úložiště vkládat, či pomocí dotazů získávat, dokumenty. Přistupovat tak lze z programu napsaného v jazyce Java (přímo z crawleru tak může dojít ke vkládání dokumentů k zaindexování) či dotazovat se z klientské části této práce (pomocí jednoduchého serveru obstarávajícího dotazy uživatele).



Obrázek 5.1: Ukázka prostředí Kibana

Instalaci a zprovoznění tohoto nástroje se podrobněji věnuje sekce 8.2 na straně 50. Samotné úložiště zabere na disku počítače přibližně 45 MB, opět nepočítaje s žádnými uloženými záznamy v databázi.

Pro snazší a intuitivnější ovládání si lze stáhnout grafické rozhraní *Ki-*

bana (vyvinuté stejnou společnost), díky kterému lze snáze monitorovat stav úložiště. V případě nejasností se na stránkách projektu dají dohledat tutoriálová videa, z nichž lze snadno pochopit základní principy fungování a ovládání těchto nástrojů.

Jak vypadá uživatelské rozhraní nástroje Kibana znázorňuje obrázek 5.1. V levé části rozhraní je na výběr nekolik možných nástrojů pro analýzu a práci s daty. „Nejzajímavější“ položkou jsou pak *Dev Tools*, tedy nástroje pro vývojáře. Po jejich otevření se obrazovka rozdělí na další dvě části (viz zmíněný obrázek). V levé z těchto dvou částí jsou zadávány příkazy, které lze odesílat na databázi ke zpracování. V pravé části jsou pak odpovědi databáze, kde se zobrazují jak výsledky požadovaných změn či zápisů do úložiště, tak výsledky vyhledávání.

6 Webové technologie

Neméně důležitou částí této práce je také reprezentace stažených informací uživatelům. K tomuto účelu budou vytvořeny webové stránky, jež jsou dnes pravděpodobně nejsnazším a nejpřístupějším způsobem, jak většímu množství uživatelů zobrazit požadované informace. Takovéto stránky využívají několika základních technologií, které stručně popisuje tato kapitola na následujících řádcích.

6.1 HTML

Základním prvkem webových stránek je dnes technologie *HTML* (HyperText Markup Language). Jedná se tzv. *značkovací jazyk*, který jednotlivé webové stránky propojuje pomocí hypertextových odkazů. Jeho struktura je dána jednotlivými značkami. Historicky vychází z původního univerzálního jazyka *SGML* (Standard Generalized Markup Language) [17].

6.1.1 Historie

S postupným rozšiřováním HTML přišly také standardy (které však pro chod webu nejsou vynucovány, což může přinášet i komplikace), pomocí kterých by se měl tento jazyk řídit. Vývoj postupně prošel několika různými verzemi, kdy nejrozšířenější verzí zůstala na velmi dlouho dobu verze 4.01. Ta byla vydána v roce 1999 a přetrvala dlouhých 15 let, kvůli pozastavení vývoje standardu.

Dnes aktuální verze nese číslo 5.0, která byla oficiálně schválena v roce 2014. Ta oproti předchozí verzi přinesla několik vylepšení, včetně nových značek. Jejím hlavní cílem je lepší podpora pro přehrávání multimediálního obsahu ve webovém prohlížeči či podpora aplikací fungujících i bez internetového připojení.

Další verze HTML by podle plánu organizace *W3C* (World Wide Web Consortium – právě tato organizace stojí za aktuálním vývojem standardu HTML) měly vycházet v dvouletých intervalech, avšak skutečnost je taková, že od roku 2014 je aktuální verzí stále zmíněná verze 5.0 [16].

6.1.2 Zápis

Jak bylo již zmíněno, jedná se značkovací jazyk. Struktura dokumentu je tedy dána „skládáním“ jednotlivých značek v určitém pořadí. Tyto značky jsou většinou párové, což znamená, že každá značka je rovněž ukončena značkou stejného typu (někdy lze využít i zkráceného zápisu). Mezi takovéto dvě značky je pak vkládán požadovaný text či další značky. Zároveň mohou jednotlivé značky ve svém „těle“ obsahovat i dodatečné atributy, podrobněji definující jejich další vlastnosti.

Základními, zároveň jedinými povinnými, značkami jazyka HTML jsou *tagy* <html>, <head> a <body>, kde první z nich „uzavírá“ ty následující. Příklad uvedený níže předvede strukturu velmi jednoduchého HTML dokumentu.

```
<!DOCTYPE html>
<html>
    <head>
        <title> Titulek webu </title>
    </head>
    <body>
        <h1> Nadpis </h1>
    </body>
</html>
```

6.2 CSS

Aby byly výše zmíněné HTML weby více atraktivní pro uživatele, je vhodné využít *CSS* (Cascading Style Sheets) kódování. Jedná se o tzv. *kaskádové styly*, pomocí který lze snadno přiřazovat různé vlastnosti (nejen) HTML tagům. Dnes se jedná o další nedílnou část moderních webových stránek. Pomocí CSS lze snadno pozicovat jednotlivé prvky na webových stránkách, stejně jako jim lze přiřazovat různé grafické úpravy [5].

Vývoj CSS standardu opěd spadá pod organizaci *W3C*. Nejnovější verzí je specifikace 3.0, jež oficiálně vyšla přibližně ve stejné době jako HTML 5.0 (tedy rok 2014). Tato nejnovější verze opět cílí na různé efekty na webových

stránkách, kterými jsou například přechody barev, různé animace elementů na webu, práce s písmem atd.

Výsledná webová stránka může obsahovat několik odlišných CSS stylizací a přepínat mezi nimi „za chodu“, dle potřeby. Pro stejný obsah webu tak lze snadno vytvořit např. zobrazení na různé velikosti obrazovky, zobrazení pro tisk stránky či jiné rozvržení webu pro zrakově či jinak postižené uživatele atd.

6.2.1 CSS prakticky

CSS se zakládá na tzv. selektorech, pomocí kterých vybíráme konkrétní prvky HTML kódu. Prvky podřízené selektovanému prvku následně „dědí“ jeho vlastnosti, odtud tedy název kaskádové.

Selektory lze podrobněji specifikovat na výber prvku s určitým *ID*, tedy jedinečným identifikátorem prvku pro celý HTML dokument. V takovém případě před názvem prvku uvedeme znak „#“. Druhou možností je selektor konkrétní třídy (těch může být v dokumentu již několik), poté před názvem použijeme znak tečky („.“). V případě, že není uveden ani jedna tato specifikace, je styl aplikován na všechny prvky daného jména v dokumentu.

Po uvedeném selektoru následuje výčet vlastností prvku, uzavřený ve složených závorkách. Jednotlivé vlastnosti jsou poté zapisovány jako „vlastnost : hodnota vlastnosti“ a jsou ukončeny středníkem („;“).

Výsledný CSS dokument se všemi požadovanými selektory je poté nutné provázat s HTML dokumentem. Následující ukázka kódu předvede styl, který obarví všechny nadpisy první úrovně (<h1>) v HTML dokumentu (z ukázky v sekci 6.1.2 na straně 22) na červenou barvu a nastaví odsazení textu nadpisu z levé strany na 10 pixelů.

```
h1 {
    color: red;
    padding: 0 0 0 10px;
}
```

6.3 Bootstrap

Jedná se o sadu nástrojů, původně vyvinutých pro sociální síť Twitter, které tvoří ucelenou soustavu CSS souborů a případně i nástavbu v podobě skriptů jazyka *JavaScript* [2]. Před vývojem tohoto nástroje využívala zmíněná síť mnoho různých *knihoven*, což často vedlo k nekonzistenci kódu. V roce 2011 byl Bootstrap vydán jako open source projekt a okamžitě se stal velmi oblíbeným.

Hlavní výhodou Bootstrapu je jeho snadné užití pro tvorbu uživatelského rozhraní. Použitím tohoto nástroje lze např. velmi snadno vytvořit responsivní design. Dále se zaměřuje na různé efekty při použití tlačítek, „popup“ nabídek, nápověd k prvkům na webu atd. Jedná se tedy hlavně o jakýsi doplněk k CSS stylům, který lze velmi snadno využít a ušetřit si tak mnoho času.

Bootstrap je opět nutné provázat s HTML dokumentem. Příklad zde uveden nebude, protože Bootstrap nemá nějaká výrazná specifika ve svém zápise. Jde spíše pouze o to přiřadit HTML prvkům vybrané CSS styly, které Bootstrap poskytuje.

6.4 AngularJS

Jedná se *JavaScriptový* webový *framework* vytvořený společností Google v roce 2009. Jeho hlavním cílem je „užší propojení“ JavaScriptu a jazyka HTML. Pomocí Angularu lze velmi snadno např. skrývat jednotlivé HTML tagy, provádět řazení prvků v seznamu „v reálném čase“ atd [1].

Aktuálně je ve vývoji Angular verze 2.0, který by měl být brzy uveden. Jeho podoba má však být velmi odlišná od aktuální verze a obě tyto verze pak ani nebudou vzájemně kompatibilní.

6.4.1 Základní hierarchie

Angural slouží k oddělení zobrazovací logiky od té aplikační, využívá tedy klasického vzoru „Model-View-Controller“. Pomocí tzv. *direktiv* jsou k jednotlivým HTML tagům vkládány atributy poskytované Angularem. Tyto

direktivy obvykle začínají zkratkou „ng-“ a následuje jejich „konkretizace“. Jako příklad lze uvést atribut *ng-if*, který zobrazí či schová požadovaný prvek na základě uvedené proměnné. Konkrétní příklad vypadá následovně:

```
...
// Pokud je proměnná „includePage“ rovna 1, zobraz vnořený obsah
<div ng-if=”includePage === 1”>
    <p> Zobrazený text </p>
</div>
...
```

Aby bylo možné Angular v HTML správně využívat, musí jeho direktivy „spadat“ pod určitý *Controller* (kontroler). Ten je navíc ještě „podmnožinou“ *modulů* Angularu a definuje hranice pro tzv. *scope* proměnné. V HTML je tak nutné tuto hierarchii dodržet, nadřazeným tagům tedy přiřadit konkrétní modul a jeho „potomkům“ poté jednotlivé konrolery.

6.4.2 Konkrétní využití

Následně je nutné vytvořit JavaScriptový dokument, ve kterém se vytvoří instance modulu a k němu pak přiřadí kontolery. Zmíněné kontrolery pak již obsahují samotný JavaScript kód. Jeho základem jsou zde proměnné označené jako *$scope*. Ty slouží jako *globální proměnné* v rámci daného kontroleru (pozor, mezi jednotlivými kontrolery nejsou navzájem přístupné!) a deklarují se následovně: **$scope.includePage = 0;**. Tento příklad do proměnné scope se jménem „includePage“ přiřadí číselnou hodnotu 0. Pokud si toto spojíme s příkladem uvedeným v sekci 6.4.1, bude text mezi tagy skrytý. Hodnotu proměnné scope lze na webu (v HTML části) zobrazit i přímo, použitím zdvojených složených závorek, tedy: **{{includePage}}** (bez uvedení slova scope). Pomocí toho lze vytvořit responsivní webové stránky okamžitě reflektující akce uživatele.

Mezi dalšími direktivami lze jmenovat například *ng-model*, kterou lze přiřadit elementu zpracovávající vstup od uživatele. Proměnná přiřazená této direktivě je měněna okamžitě po akci uživatele, což se hodí zejména na *input* prvky.

Neméně důležitou direktivou je pak i *ng-click*. Ta, jak již název napovídá, zpracuje určitou akci (například spuštění nějaké funkce) pokud uživatel klikne

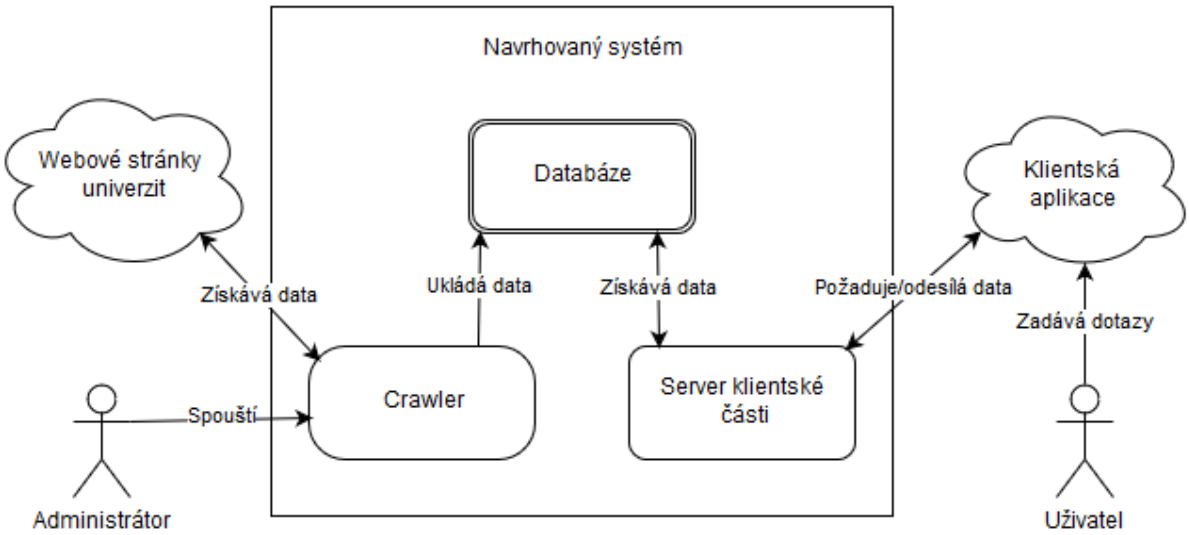
na prvek obsahující tuto vlastnost.

V neposlední řadě lze jmenovat i direktivu *ng-repeat*. Ta jednoduše iteruje přes všechny prvky zadaného objektu (ten musí být iterovatelný) a dokáže tak „opakovat“ stejnou posloupnost tagů několikrát za sebou, bez nutnosti znát jejich přesný počet. Jako příklad této direktivy poslouží kód níže. Ten provede iterování nad objektem **responseData** a zobrazí pod sebou vypsaná jména předmětů, která jsou obsažena v tomto objektu.

```
...
<div ng-repeat=”subject in responseData”>
    <p> {{subject.name}} </p>
</div>
...
```

7 Návrh systému

V této kapitole bude stručně popsán navrhovaný systém, jaké součásti bude využívat a jakým způsobem bude provozován. Základní strukturu navrhovaného systému popisuje následující obrázek (obr. 7.1):



Obrázek 7.1: Navrhovaný systém

Pokud začneme systém popisovat od jeho středu, tedy od databáze, je nutné tuto databázi nejprve naplnit. O tuto činnost se bude starat administrátor, který by měl být alespoň zhruba seznámen s principy fungování tohoto systému. Hlavním důvodem, proč by práci administrátora neměl zastávat úplný laik, je ten, že může nastat situace, kdy některá z univerzit pozmění či dokonce úplně vymění své webové stránky. V takovém případě bude nutný zásah do kódu crawleru, který je naprogramován „na míru“ jednotlivým stránkám univerzit a změny jejich struktury nedokáže reflektovat. Takováto oprava kódu crawleru není složitá záležitost, avšak měl by ji provádět člověk seznámený s tímto projektem, respektive člověk schopný provést programové úpravy crawleru. Pokud nebudeme předpokládat žádné výjimky, stačí spustit program crawleru a jednoduše, pomocí pár kliknutí, stáhnout potřebná data. Ta se pak nahrají do připravené databáze stisknutím dalšího tlačítka.

Na stejné pracovní stanici, na které je nainstalována a spuštěna databáze (a na které bude pravidelně spouštěn i crawler), poběží rovněž i server pro obsluhu klientských požadavků. Administrátor bude opět dohlížet na

jeho bezchybný chod. Server funguje velmi jednoduše – od uživatele přijme požadavek ve formátu JSON dokumetu (důvod výběru právě JSON dokumentů vysvětluje sekce 5.1.2 na straně 16, jeho struktura je blíže popsána v sekci 7.3 na straně 30), který bude obsahovat potřebné parametry. Na základě zaslaných parametrů poté proběhne komunikace s databází, která případné výsledky předá zpět serveru. Ten z nich následně složí odpověď (opět ve formátu JSON dokumentu) pro webové rozhraní, a odešle ji pro zobrazení uživateli.

Uživatel zde zastupuje „externí roli“, kdy se systémem komunikuje pouze pomocí webové aplikace (ta je rovněž součástí serveru na pracovní stanici a je veřejně přístupná). Toto rozhraní nabídne velmi jednoduchý a na ovládání intuitivní design. Uživatel bude do připraveného pole zadávat své dotazy, které bude tlačítkem odesílat na server pro zpracování. Aplikace rovněž nabídne několik možností, jak výsledná data filtrovat. Po obdržení odpovědi ze serveru jsou výsledky uživateli zobrazeny ve stejné části webové aplikace, ve které odesílal své dotazy.

7.1 Crawler

Jednou z hlavních součástí systému bude bezpochyby webový crawler. Ten bude napsán v jazyce Java a bude ovládán přímo administrátorem. Zhotoven bude jako samostatně spustitelná aplikace. Aby však bylo možné nahrávání dat do databáze, musí být spouštěn na stejné pracovní stanici, na které je spuštěna a připravena i zvolená databáze. Jeho hlavním účelem bude jednorázové stažení dat ze stránek vybraných univerzit a následné nahrání těchto dat do databáze, respektive vyhledávacího enginu.

Crawler bude spouštěn jako samostatná aplikace obsahující jednoduché, na ovládání nenáročné, grafické rozhraní. Bude nabízet stažení dat pro každou univerzitu nezávisle na sobě, aby při případné chybě stahování dat z jedné z nich nedošlo k zastavení celého procesu. Data budou stahována jednou ročně, vždy v době, kdy dochází k obměně vyučovaných předmětů. Vyučované předměty se pak v průběhu školního roku na vysoké škole nemění. Podstatné pak je, aby byla data stahována vždy za oba semestry daného roku zároveň.

Hlavním předpokladem pro správnou funkcionalitu crawleru je zde fakt, že se struktura webových stránek univerzit nezmění. V opačném případě bude vždy nutné crawler upravit takovým způsobem, aby odpovídal aktuální

struktuře webů. Tento problém platí jak pro ručně psaný crawler od základu, tak i případné využijí některého z hotových nástrojů – u těch je taktéž nutné upravovat šablonu, jakým způsobem mají být data vyhledávána.

Aby bylo možné jednotlivé předměty porovat, je nutné získat z rozdílných webových stránek univerzit jednotné informace. Jak bylo dříve zmíněno, v kapitole 3 na straně 4, ne všechny informace jsou vždy dostupné, případně se mohou stejné informace na webu nacházet pod různými názvy. Ve výsledku je však nutné uchovávat minimálně následující informace, podle kterých bude uživatel schopen porovnávat dva různé předměty:

- název univerzity
- název fakulty
- název katedry
- název předmětu
- zkratka předmětu
- anotace k předmětu
- osnovy přednášek předmětu
- počet kreditové dotace, kterou předmět obdržel
- URL odkaz na předmět, odkud byly informace staženy

Anotací předmětu se zde rozumí shrnutí toho, jakou oblastí se daný předmět zaobírá. Osnovami je pak myšlen již konkrétnější popis, jakému tématu se například věnují jednotlivé přednášky předmětu.

7.2 Úložiště

Úlohu úložiště v této práci zastane dříve zmíněný nástroj ElasticSearch (sekce 5.2.3 na straně 18). Jeho snadná použitelnost, dostupnost a rychlost z něj udělaly vhodného kandidáta na pozici datového úložiště.

V prostředí programovacího jazyku Java probíhá komunikace s úložištěm pomocí poskytovaného *API*. K tomuto účelu byl naprogramován server klientské části (sekce 7.3 níže). Z tohoto serveru obdrží databáze požadavky sestavené na základě uživatelem odeslaných dat. Na základě těchto požadavků pak databáze sestaví odpověď s výsledky, kterou odesílá zpět na server. Odpověďmi databáze jsou dokumenty formátu JSON, které splňují požadavky vyhledávání a které jsou řazeny podle relevance vyhledávaných záznamů. Tyto JSON dokumenty lze pak již velmi snadno zobrazit v klientské části aplikace.

Přes zmíněné API probíhá s úložištěm komunikace i v samotném crawleru, při požadavku uložit či smazat data v databázi.

7.3 Server klientské části

Pro tuto část práce byl zvolen server *TomCat* verze 7, který bude spuštěn na stejné pracovní stanici jako dříve zmíněná databáze. Server bude *mapován* pod určitým jménem jako tzv. *servlet* a bude tak na určité adrese naslouchat příchozím dotazům od uživatelů.

Na tento server budou jednotliví uživatelé zasílat své dotazy, na jejichž základě server vytvoří a odešle dotaz na vyhledávání v databázi. Odpověď od databáze pak zašle zpět uživateli.

Příchozí dotazy bude zpracovávat z datového formátu JSON, konkrétně musí obdržet několik parametrů, kterými jsou:

- hledaný výraz
- pole hodnot pravda/nepravda odpovídající volbě uživatele, ve kterých informacích o předmětu bude zadaný výraz vyhledáván
- pole dvou číselných hodnot udávajících rozsah kreditního ohodnocení předmětu (číslo 7 je zde bráno jako „maximum“ a převedeno na číslo 100, aby byl zahrnut maximální rozsah kreditů)
- pole hodnot pravda/nepravda, které určí, mezi jakými univerzitami bude dotaz vyhledáván

Pokud uživatel odešle dotaz, ve kterém jsou všechny dostupné možnosti nastavení zaškrtnuty a kreditní rozsah je nastaven od nuly do „maxima“, vypadá výsledný JSON dokument, respektive dotaz na databázi, pro vyhledávané spojení „programování v jazyce C“ následovně:

```
{ ”searchInput”:”programování v jazyce C”, // Vstup od uživatele
”searchFields”:[0:true, 1:true, 2:true], // Oblast hledání
”creditsRange”:[0:0, 1:7], // Rozsah kreditů
// Výčet vybraných univerzit
”searchUni”:[0:true, 1:true, 2:true, 3:true, 4:true, 5:true, 6:true, 7:true,
8:true, 9:true],
// Posun hledaných výsledků v databázi
”queryFrom”:0 };
```

Odpověď od databáze server převezme a odešle do webové aplikace jako pole JSON dokumentů. Tyto dokumenty obsahují informace o jednotlivých předmětech (které konkrétní informace to jsou popisuje sekce 7.1 na straně 28). Pokud je nějakým způsobem vyvolána výjimka během provádění dotazu na databázi (což může být způsobeno například odesláním požadavku se špatným počtem parametrů – tedy pokud se někdo snaží „podstrčit“ serveru neplatný dotaz), vrací server uživateli chybový kód a ve webové aplikaci dojde k přesměrování na část webu s chybovými hlášeními.

7.4 Klientská aplikace

Klientská, respektive webová, aplikace bude zhotovena jako webová stránka, na níž uživatel zadává dotazy, a po jejich odeslání jsou mu zobrazeny výsledky těchto dotazů.

Aplikace bude stavěna na kódovacím jazyce *HTML* ve verzi 5. Pro stylování webu bude využito technologie *CSS* verze 3. Dále zde bude využito skriptovacího jazyka *AngularJS* a knihovny zvané *Bootstrap* (ta např. usnadní práci s *layoutem* webu, který tak může být snadno postaven pro přívětivější zobrazení i pro mobilní zařízení – viz sekce 6.3 na straně 24).

Nejvýraznějším prvkem webu by mělo být vyhledávací pole, do kterého bude zadáván text ve formě vhodných dotazů. Ty mohou být i víceslovné, avšak vždy by se měly soustředit pouze na konkrétní „téma“. Uživatel by měl

zadávat dotazy takovým způsobem, aby bylo možné danou frázi vyhledat v databázi v prostém textu. Jako příklad lze uvést dotaz „počítačové sítě“. Uživatel tak může očekávat, že mu budou nabídnuty předměty, které souvisejí s tématem počítačů a počítačových sítí, respektive obsahují tato slova ve svém názvu či popisu.

Uživatel by neměl při odesílání jednoho dotazu zadávat více různých klíčových výrazů. Důvodem je fakt, že databáze implicitně, z důvodu rychlosti, vrací maximálně vždy jen 10 nejlépe odpovídajících výsledků daného dotazu. Toto číslo je možné navýšit, ale není to doporučováno, proto bude ponecháno na této hodnotě. Díky tomu, odešle-li uživatel více dotazů postupně za sebou, bude uživateli vlastně nabídnuto více možností pro každý z hledaných výrazů, což je pro uživatele pochopitelně výhoda.

Dále by měl web umožňovat, pomocí ovládacích prvků, nastavit parametry pro přesnější vyhledávání požadovaných dotazů. Vyskytovat by se zde měly prvky, pomocí kterých si uživatel vybere, mezi kterými informacemi ke každému předmětu chce zadaný výraz vyhledávat. Tyto možnosti byly vybrány tři, konkrétně: „název předmětu“, „anotace předmětu“ a „osnovy předmětu“. Pokud uživatel neurčí jinak, bude klíčové slovo hledáno mezi všemi třemi možnostmi.

Další vhodnou možností je také výběr kreditního ohodnocení předmětů. Uživatel by měl mít možnost zvolit si rozsah, díky kterému dostane přesnější výsledky. Rozsah by měl být volitelný od 0 do 6 a více kreditů. Jako „maximum“ byla zvolena hodnota 6+, což značí předměty s kreditním ohodnocením 6 a více. Z osobní zkušenosti se však mnoho předmětů s takovýmto kreditním ohodnocením nevyskytuje.

Výsledky hledání je taktéž vhodné upřesnit výběrem konkrétních univerzit, na kterých budou předměty vyhledávány. Web by tedy měl obsahovat prostředky, pomocí kterých si uživatel jednoduše vybere, zda ho zajímají pouze některé konkrétní univerzity, či si přeje předměty vyhledávat ve všech nabízených. Toto je rovněž další část kódu, který bude nutno administrátorem upravit, pokud bude někdy v budoucnu do této práce přidáno více univerzit.

Výsledky odeslaných dotazů by pak měly být zobrazeny v rámci stejné webové stránky, na které se nachází prvky nastavení a formulář pro odeslání dotazu. Zobrazeny budou jako seznam názvů předmětů, které odpovídají zadanému dotazu. Pro přehlednost bude dobré nejprve zobrazovat pouze názvy, po jejich rozkliknutí pak dodatečné informace o daném předmětu (o jaké in-

formace se jedná viz kapitola 7.1 na straně 28).

Na základě zobrazených datailů se uživatel bude moci rozhodnout, zda je pro něj předmět zajímavý. Web by tak měl poskytovat možnost (např. ve formě tlačítka) přidat si vybraný předmět mezi „oblíbené“. Obdobným způsobem by pak měl jít předmět z „oblíbených“ i odebrat.

Poslední důležitou součástí webu bude počítadlo „oblíbených“ předmětů. K tomu se rovněž váže možnost přepínat se na druhou část webu, na které budou vybrané předměty zobrazeny. V této druhé části by uživatel měl mít možnost znovu procházet vybranými předměty s volbou je, dle uvážení, odebírat ze seznamu. Zároveň by se v této části měla nacházet souhrnná informace o tom, kolik předmětů z jaké univerzity si uživatel vybral. Na základě výběru by poté byla zobrazena, například ve formě loga a názvu univerzity, ta vysoká škola, pro kterou si zvolil nejvíce „oblíbených“ předmětů.

8 Realizace systému

V této kapitole bude podrobněji popsána praktická realizace této práce. Všechny části popsané v kapitole 7 (Návrh systému) na straně 27 zde budou jednotlivě rozebrány a ke každé z nich podrobně popsáno, jakým způsobem byla daná část zhotovena.

Samotný programový kód práce je hojně komentován v anglickém jazyce, s jeho pochopením jiným programátorem a následným rozšiřováním by neměl nastat žádný problém.

8.1 Crawler

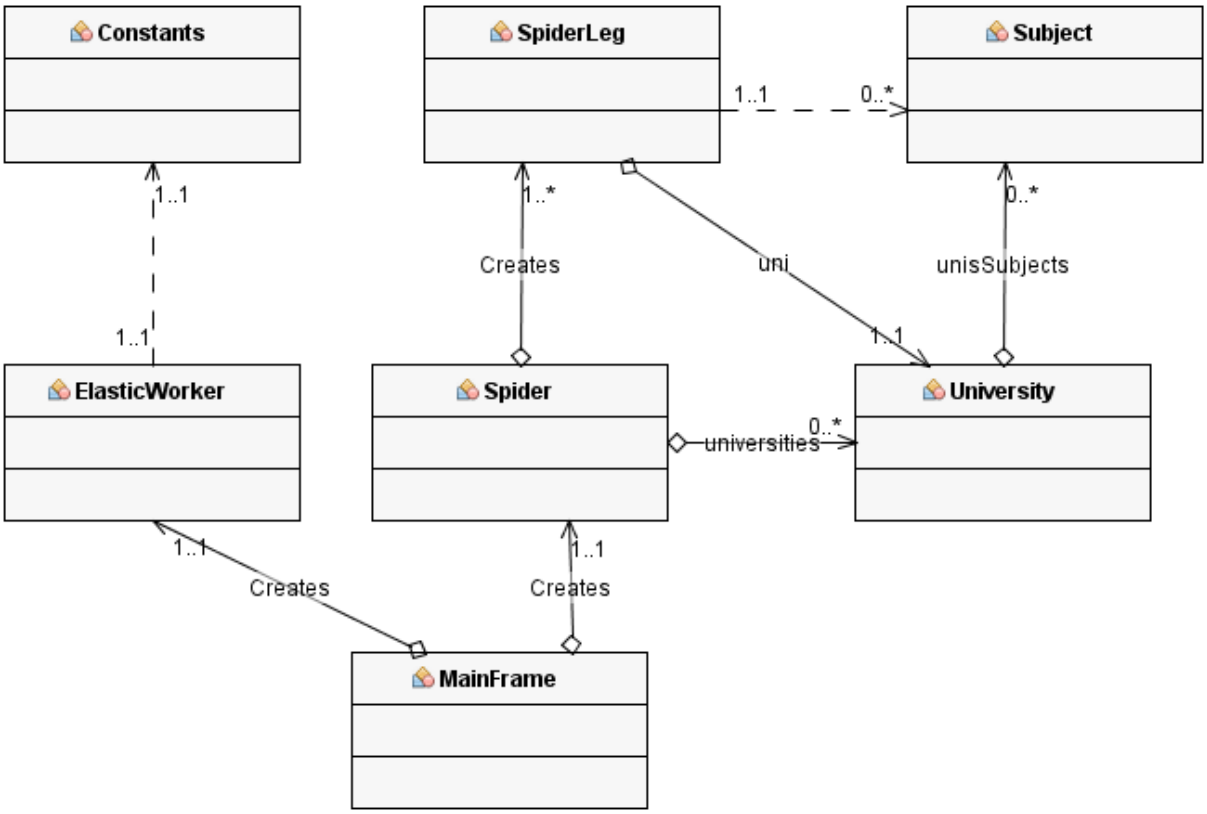
Jak bylo popsáno již dříve, crawler slouží jako výchozí bod této práce. Jeho hlavním účelem je stažení požadovaných informací z vybraných webových stránek. K vývoji systému bylo použito vývojového prostředí NetBeans IDE verze 8.0.2. Tématu crawlingu se věnuje i zdroj [35].

8.1.1 Technické detaily

Tato část systému vyžaduje ke svému spuštění platformu *Java SE* verze 8, respektive *JDK verze 1.8* pro vývoj. Právě minimálně tato verze je totiž vyžadována datovým typem *TransportClient*, který v crawleru slouží pro komunikaci s databází.

Pro snazší práci při crawlingu webových stránek byla využita knihovna *Jsoup*. Ta poskytuje rozhraní, které nabízí mnoho užitečných metod, které je dostupné po připojení této knihovny k projektu. Další použitou knihovnou je pak *Client* od ElasticSearch, které slouží pro komunikaci s rozhraním úložiště ElasticSeach. Právě tato knihovna požaduje využití zmíněného JDK verze 1.8.

Program crawleru je rozdělen celkem do 7 tříd. Jejich diagram si lze prohlédnout na obrázku 8.1 na straně 35. Na něm je uvedeno vzájemné provázání tříd, bez bližších detailů k jednotivým třídám. Ty budou uvedeny na obrázcích zvlášť, při detailnějším popisu tříd.

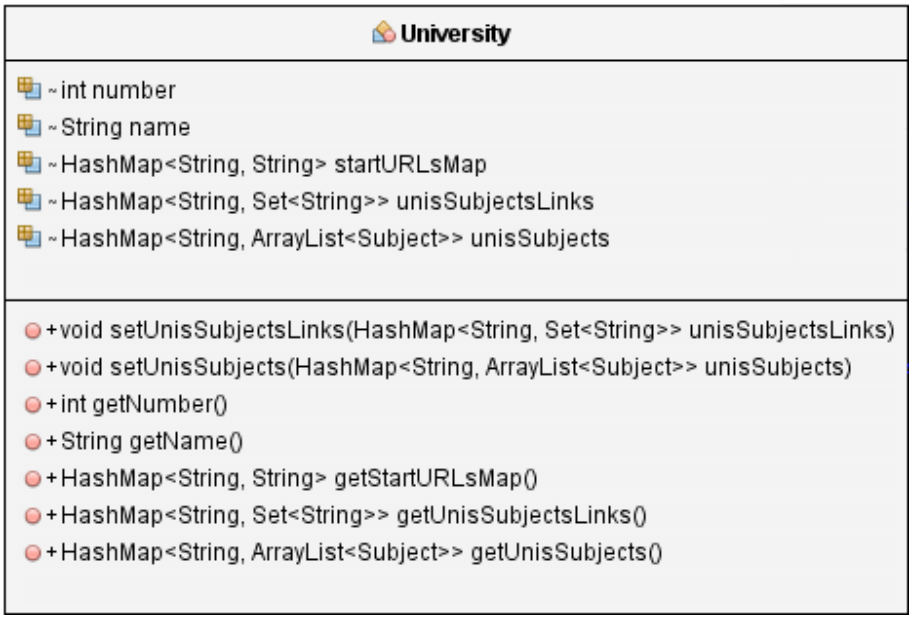


Obrázek 8.1: Diagram tříd crawleru

Dvě z nich slouží jako „šablona“ pro vytvoření objektů univerzit (třída *University* – detaily viz obrázek 8.2 na straně 36) a jednotlivých předmětů (třída *Subject* – obr. 8.4 na straně 37, rozdělený do dvou sloupců pro lepší přehlednost). Ty obsahují nezbytné proměnné pro vytvoření instance těchto tříd a metody *get* a *set* pro nastavení, respektive získání, jejich proměnných. Třída *Universities* pak navíc obsahuje kód pro vytvoření konstruktoru třídy. Jeho atributy jsou číslo univerzity (0 až 9, přiřazené na základě abecedně seřazených jmen univerzit), jméno univerzity a objekt *HashMap*, jehož klíčem je jméno fakulty, hodnotou pak počáteční URL pro start crawleru.

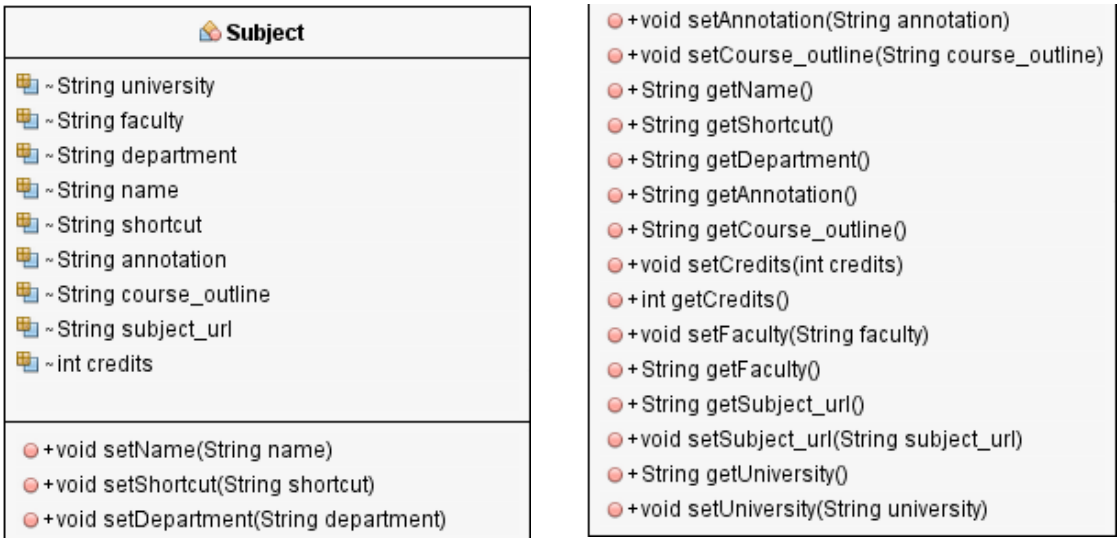
Další třída pak slouží pouze pro „zpřehlednění“ kódu. Její název je *Constants* (obrázek 8.3 na straně 36) a obsahem jsou pak proměnné využívané na více místech programu, v ostatních třídách. Zároveň také obsahuje metody *get* a *set* pro nastavení proměnných, které jsou využívány pro přístup k DB – původně byly tyto hodnoty konstantní, na základě připomínek byla přidána možnost načíst je z konfiguračního souboru, pokud by bylo potřeba změny.

Zbylé 4 třídy již slouží pro vykonávání samotné činnosti crawleru a budou podrobněji popsány níže, případně se lze o jejich účelu dočíst v komentářích kódu programu.



Obrázek 8.2: Třída University

Obrázek 8.3: Třída Constants



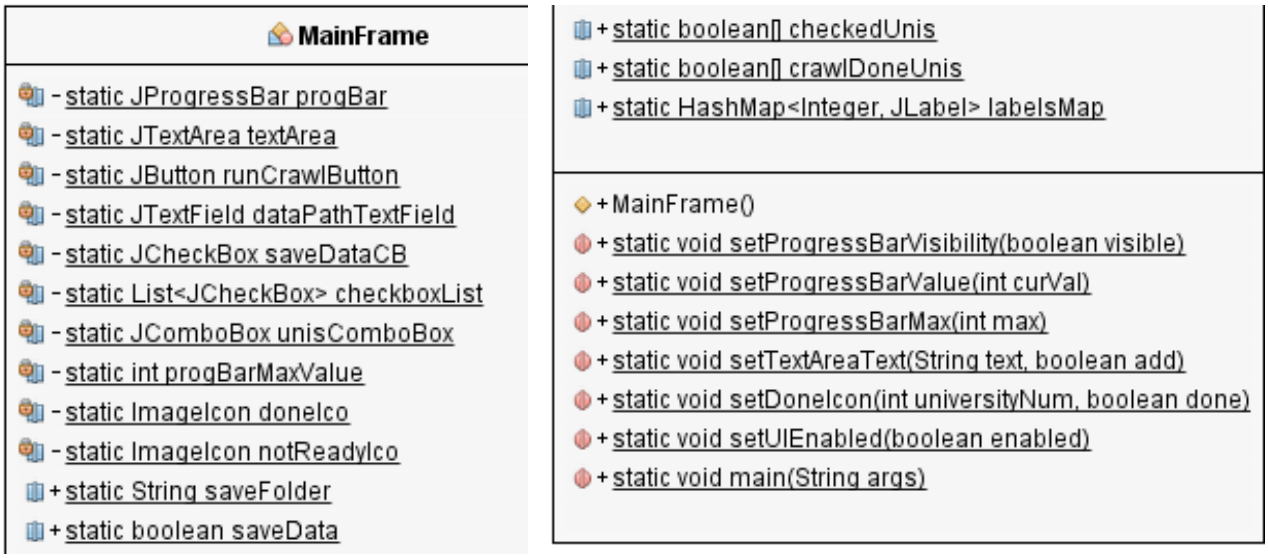
(a) (b)

Obrázek 8.4: Třída Subject

8.1.2 Třída MainFrame

Pomocí této třídy dojde ke spuštění celého crawleru. Její proměnné a metody si lze prohlédnout na obrázku 8.5 na straně 38. Nejdůležitější metodou je ***public static void main(String args[ ])***, ve které dojde k vytvoření a spuštění *GUI* (Graphic User Interface – grafické uživatelské rozhraní). Přes toto rozhraní pak administrátor pohodlně „nastavuje“ a spouští crawling dat, případně pracuje s databází.

Při tvorbě crawleru bylo dbáno na to, aby samotná část, která se stará o získávání informací ze stránek, byla co nejméně závislá na tomto grafickém rozhraní, respektive třídě. To znamená, aby bylo možné GUI kdykoliv odstranit či nahradit bez ohledu na funkcionalitu crawleru. Tohoto je v práci docíleno jen částečně. Například seznam univerzit, které byly při spuštění crawlingu zaškrtnuty, je uchováván v této třídě v proměnné ***checkedUnis*** ve formě pole hodnot *boolean*. Dalším příkladem propojení této třídy s ostatními je pak jejich závislost na grafickém prvku *JTextArea*, který zobrazuje hlášení vyvolaná při crawlingu či práci s databází. K výpisu hlášení pak slouží metoda ***public static void setTextAreaText(String text, boolean add)***. Jejími parametry jsou textový řetězec obsahující text hlášení a *boolean* hodnota určující, zda se má text přidat k již existujícímu (v případě hodnoty *true*) či přemazat dosavadní text.



(a) (b)

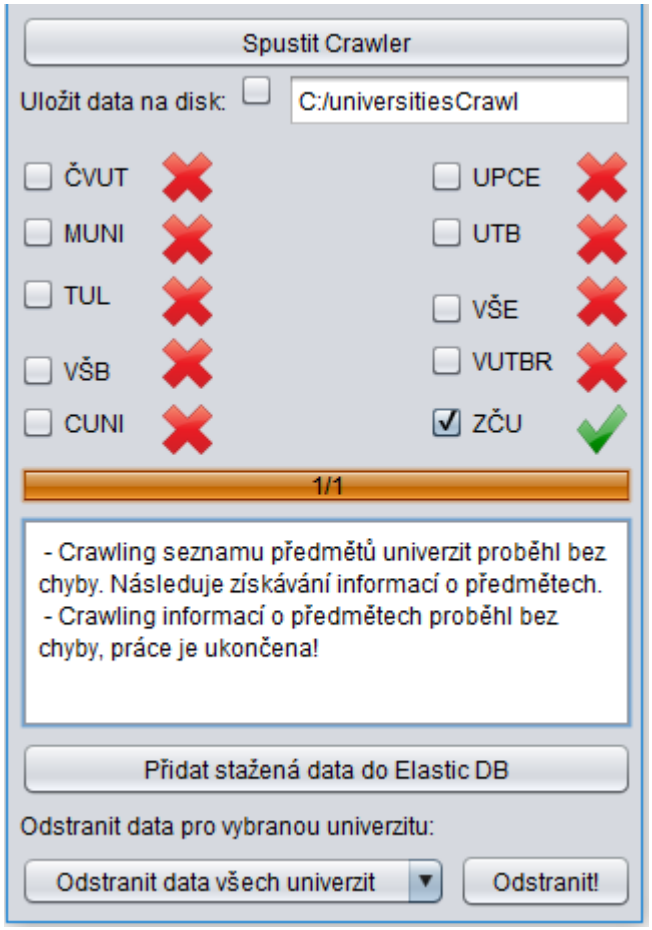
Obrázek 8.5: Třída MainFrame

GUI bylo navrženo s ohledem na co nejjednodušší manipulaci. Jeho vzhled si lze prohlédnout na obrázku 8.6 níže. Základem je výčet vybraných univerzit, pro které je možno spustit crawling. Tento výčet je vyhotoven pomocí zaškrtávacích tlačítek (kdy zaškrtnutý stav znamená, že daná univerzita bude zpracována crawlerem) s příslušným názvem univerzity a obrázkem znázorňujícím, zda crawlink pro danou univerzitu proběhl v pořádku (zelená „fajfka“) či nikoliv (červený „křížek“). Jednotlivá tlačítka jsou uložena v proměnné ***checkboxList***, obrázky pak v proměnné ***labelsMap*** – ty jsou zhotoveny jako pozadí prvků grafického prostředí *Swing – Label*.

V horní části rozhraní se nachází tlačítko, kterým dojde ke spuštění samotného procesu crawlingu. Ten již obstarává třída ***Spider*** (její popis v sekci 8.1.3 na straně 41).

Pod zmíněným tlačítkem se dále nachází další ze zaškrtávacích komponent – tato konkrétně slouží pro volbu uložení stahovaných dat na disk počítače. Tuto možnost stačí zaškrtnout (implicitně zvolena není, tlačítko je provázáno s proměnnou ***saveData***) před spušněním crawleru a do textového pole (prvek *JTextField*) zadat cestu, kam mají být data uložena. Tato cesta je uchována v proměnné ***saveFolder***, jejíž implicitní hodnota je nastavena na „C:/universitiesCrawl“. V případě zaškrtnutí této možnosti jsou data ukládána ve třídě ***SpiderLeg***, po jejich získání. Zde je tedy opět další

propojení třídy *MainFrame* a třídy *SpiderLeg*, pomocí zmíněné proměnné *saveData*. Možnost uložení dat je zde spíše jako volba „navíc“, primárně zde jde o uložení dat do databáze. Ukládání na disk počítače se však může hodit například pro archivaci získaných dat.



Obrázek 8.6: Ukázka GUI crawleru (spuštění pro univerzitu ZČU)

Ve spodní části GUI se pak nachází již dříve zmíněný prvek *JTextArea*, který slouží k výpisu informací o činnosti crawleru. Pod tímto prvkem se nachází tlačítko pro přidání stažených informací do databáze, s textem „Přidat stažená data do Elastic DB“. To je nutné použít až po získání dat, která jsou stažena pomocí crawleru.

Posledními ovládacími prvky jsou zde rolovací menu (zhotovené pomocí prvku *JComboBox*), ve kterém je možnost vybrat konkrétní univerzitu (či všechny najednou). Následným stiskem tlačítka s popisem „Odstranit!“ lze smazat data z úložiště pro zvolenou univerzitu (případně univerzity). Po

stisku zmíněného tlačítka se program uživatele ještě dotáže (skrze dialog generovaný pomocí *JOptionPane*), zda si uživatel opravdu přeje smazat vybraná data. To zabrání nechtěnému vymazání. Po souhlasu je vytvořena nová instance třídy ***ElasticWorker***, které je předána hodnota *false* (značící mazání dat) a index zmíněného comboBoxu. Ten určí buď konkrétní univerzitu, či v případě čísla 0 pak mazání všech dat.

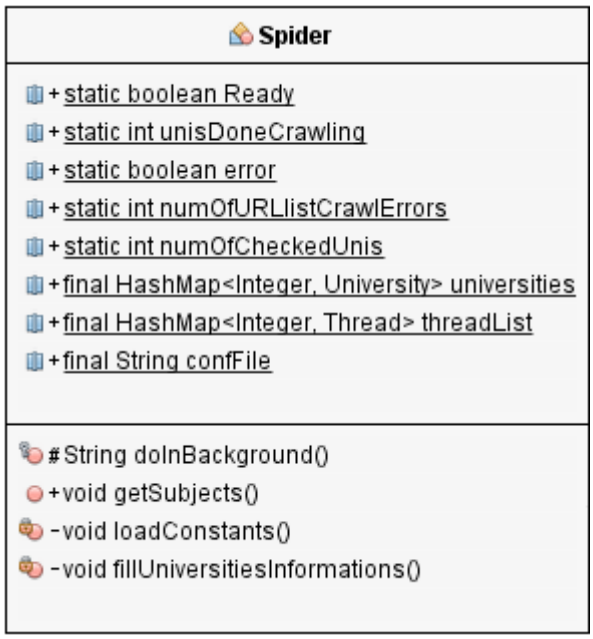
Dále pak třída *MainFrame* obsahuje metody, které slouží pro manipulaci s GUI. Mezi ty patří ***setProgressBarVisibility***, která nastavuje viditelnost ukazatele, kolik univerzit ze všech zvolených již ukončilo crawling. Dále pak metody ***setProgressBarValue***, respektive ***setProgressBarMax***, pro nastavení hodnot zmíněného ukazatele. Metoda ***setTextAreaText*** slouží pro nastavení textu příslušnému *JTextArea* prvku GUI. Metoda ***setDoneIcon*** na základě parametru *universityNum* (tedy čísla univerzity) a jedné *boolean* hodnoty nastaví obrázek na pozadí pro *Label* příslušné univerzity (ten je čerpán z *resource* projektu). Poslední, dosud nejmenovanou, metodou je pak ***setUIEnabled***, díky které lze jednoduše nastavit ovládací prvky GUI na *enabled*, respektive *disabled*.

Při spuštění programu dojde nejprve k inicializaci potřebných proměnných (jejich výčet viz obr. 8.5 na straně 38) a vytvoření instancí jednotlivých ovládacích prvků. Po stisku spuštění crawleru pak proběhne kontrola, které všechny univerzity byly vybrány (tím je nastavena proměnná *progBarMaxValue* na příslušnou hodnotu) a případné vytvoření složky, pokud si uživatel přeje uložit data na disk. Poté, v případě že je vybrána alespoň jedna univerzita (*progBarMaxValue* je větší než 0), jsou nastaveny obrázky příslušné jednotlivým univerzitám, které indikují, že crawling ještě nedoběhl. Dále se „zneaktivní“ GUI a dochází k vytvoření instance třídy *Spider*, která se již stará o řízení crawlingu.

Metoda ***deleteUniversityBActionPerformed***, která obsluhuje stisknutí tlačítka pro přidání dat do databáze, se postará o vytvoření instance třídy *ElasticWorker* (podrobněji v 8.1.5 na straně 48). Podobně funguje i metoda obstarávající mazání dat z databáze – vytváří instanci stejné třídy (pouze s rozdílem prvního parametru, který je v případě mazání nastaven na *false*).

8.1.3 Třída Spider

Tato třída slouží ke spuštění samotného procesu crawlingu informací. Jsou vytvořeny a spuštěny jednotlivá vlákna, která poté získavají požadovaná data. Kód zmíněných vláken je uložen ve třídě *SpiderLeg* (podrobněji viz sekce 8.1.4 na straně 43). Proměnné a metody třídy *Spider* pak zobrazuje obrázek 8.7 níže.



Obrázek 8.7: Třída Spider

Třída *Spider* rozšiřuje třídu ***SwingWorker***, díky čemuž je při jejím spuštění vytvořeno vlákno „na pozadí“. To umožňuje ovládání GUI crawleru i při spuštění samotné akce crawlingu, nedochází tak k jeho „zamrznutí“, pokud činnost probíhá.

Při vytvoření instance třídy a jejím spuštění tak dojde k vyvolání metody ***doInBackground***. Ta volá metody ***loadConstants*** a ***getSubjects***. První jmenovaná slouží k načtení dat z přiloženého konfiguračního souboru, pomocí nichž jsou nastaveny konstanty pro připojování k úložišti. Tyto konstanty (adresa, port a index) se nacházejí ve třídě *Constants*.

Druhá zmíněná metoda, *getSubjects*, nejprve nastaví proměnnou ***error*** na hodnotu *false* (to značí, že zatím nenastala žádná chyba) a proměnnou ***unisDoneCrawling*** na hodnotu 1 (počet univerzit s úspěšným crawlin-

gem). Poté vyvolá metodu ***fillUniversitiesInformations***. Ta naplní *HashMapu* ***universities*** potřebnými daty pro crawling (klíčem je číslo univerzity, hodnotou pak instance třídy *University*, která v sobě obsahuje název univerzity a *HashMapu* s URL, ze kterých má začít crawler stahovat).

Ukázku, jakým způsobem aktuálně zmíněná metoda *fillUniversitiesInformations* plní mapu informacemi, si lze prohlédnout v následujícím kódu:

```
...
// Vytvoří novou mapu pro seznam URL - klíčem je název fakulty
facultyAndStartURL = new HashMap<>();
// Vloží do mapy URL pro FAV
facultyAndStartURL.put("FAV",
                       "http://ects.zcu.cz/browser/FAV?lang=cs");
// Vloží do mapy URL pro FEK
facultyAndStartURL.put("FEK",
                       "http://ects.zcu.cz/browser/FEK?lang=cs");
// Do mapy univerzit vloží číslo a novou instanci třídy univerzity
universities.put(9, new University(9, "ZCU", facultyAndStartURL));
...
```

Výše uvedený příklad ukazuje, že do mapy *facultyAndStartURL* jsou vkládány dvě fakulty, respektive „startovní“ URL. V rámci této práce se však pracuje pouze s uvedenou fakultou *FAV*. Ukázka ale demonstruje fakt, že práce již počítá s následným rozšiřováním a je na něj v tomto ohledu připravena.

K výše uvedenému kódu je ještě nutno dodat, že čísla při vytváření nové instance univerzity musí korespondovat s označením zaškrtávacích tlačítek v GUI crawleru. Tedy tak, že v poli zaškrtnutých univerzit odpovídá devátá pozice univerzitě ZČU.

Poté jsou již vytvářena nová vláka třídy *SpiderLeg*, v závislosti na tom, které univerzity byly v GUI vybrány. Zmíněná vlákna jsou vlastně zakládána dvakrát. Nejprve dochází ke crawlingu seznamu URL všech předmětů, které poté budou procházeny (vlákno založeno a spuštěno s parametrem *true*). Pomocí funkce *join* jsou poté všechna vlákna synchronizována, tzn. kód na další vykonávání činnosti se pozastaví, dokud všechna vytvořená vlákna neukončí svou činnost.

Až poté dochází k založení další sady vláken, tedy pokud je proměnná

numOfURLlistCrawlErrors menší než počet uživatelem zvolených univerzit (proměnná ***numOfCheckedUnis***), tzn. pokud se podařilo získal alespoň jeden seznam předmětů ze všech zvolených univerzit. V těchto dalších vláknech již probíhá stahování dat konkrétních předmětů (vlákno založeno a spuštěno s parametrem *false*). Poté následuje synchronizace všech dříve spuštěných vláken.

Pokud v této části crawlingu dojde k nějakému problému (proměnná *error* má hodnotu *true*), je uživatel informován v příslušné části grafického rozhraní. Pokud problém nenastal, práce je u konce a dochází k „odblokování“ GUI pomocí metody *setUIEnabled* ze třídy *MainFrame* (zde dochází k provázání třídy obsluhující GUI a crawleru).

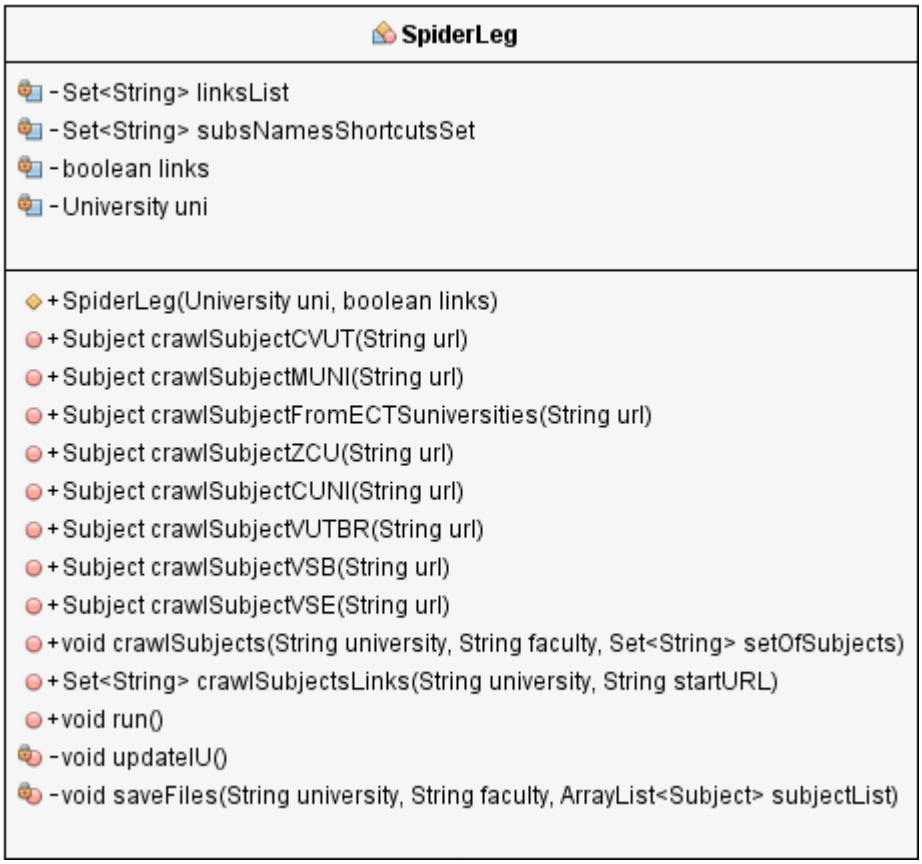
8.1.4 Třída SpiderLeg

Jedná se o třídu, ve které probíhá samotný crawling informací z webových stránek. Návody na manipulaci s knihovnou *jsoup* byly čerpány ze zdrojů [18] a [19].

Tato rozšiřuje třídu *Thread* a je tedy spouštěna jako nové vlákno systému. Seznam jejích proměnných a metod si lze prohlédnout na obrázku 8.8 na straně 44. K vytváření nových instancí této třídy slouží *konstruktor* ***public SpiderLeg(University uni, boolean links)*** – jeho prvním parametrem je instance třídy univerzity, druhým pak proměnná typu *boolean*, která určí, zda má dojít ke crawlingu seznamu předmětů (*true*) či crawlingu již konkrétních dat o předmětech (*false*).

Metoda ***public void run()*** je vyvolána ihned po spuštění vlákna. V ní jsou na základě proměnné ***links*** (nastavené v konstruktoru třídy) získávány seznamy všech předmětů (respektive jejich „startovních“ URL) či konkrétní informace o předmětech (v případě, je-li *links* nastaveno na hodnotu *false*).

V případě crawlingu seznamu předmětů je, z univerzity předané v konstruktoru třídy, získána *HashMapa* obsahující počáteční URL pro každou z uložených fakult (***uni.startURLsMap*** – aktuálně pouze jedna fakulta pro každou univerzitu). O vytvoření a naplnění této mapy se zmiňuje sekce 8.1.3 výše. Z mapy jsou iterací nad všemi jejími hodnotami získány konkrétní počáteční URL pro crawling. Ty jsou spolu s názvem univerzity předány jako parametr metodě ***public Set<String> crawlSubjectsLinks(String university, String startURL)***.



Obrázek 8.8: Třída SpiderLeg

Crawling seznamu předmětů

Metoda *crawlSubjectsLinks* se přímo připojuje na zadané URL, na němž získá všechny odkazy. K tomuto kroku využívá knihovny *Jsoup* jmenované v sekci 4.2.3 na str. 14. Kód pro připojení a získání dokumentu vypadá následovně:

```
// startURL je adresa, ke které se připojuje
// USER_AGENT pak metainformace pro weby, identifikující crawler
Connection connection = Jsoup.connect(startURL)
                        .userAgent(USER_AGENT);
// timeout určí, po kolika ms vyprší pokus o získání dokumentu
Document htmlDocument = connection.timeout(
            Constants.jsoupConnectionTimeout).get(); ...
```

Pokud z nějakého důvodu selže připojení na dané URL (či dojde při crawlingu seznamů k chybě), je způsobená výjimka „odchycena“ blokem *catch*, ve kterém dochází k informování uživatele crawleru, že nastala chyba (odesláním textu do GUI). Dále je proměnná *numOfURLlistCrawlErrors* zvýšena o 1, což indikuje minimálně jednu chybu. Hodnota proměnné *error* je rovněž nastavena na *true*.

Pokud připojení proběhne v pořádku, pokračuje program dále, získáním všech odkazů (HTML element <a>) na předaném URL:

```
Elements links = htmlDocument.select("a");
```

Následuje *switch*, ve kterém je na základě předané proměnné *university* (řetězec se zkratkou jména univerzity) vybrána příslušná „větev“ programu, ve které crawling pokračuje. Crawler je zde naprogramován „na míru“ jednotlivým webovým stránkám a vyhledává tak konkrétní výrazy. Pokud tedy dojde ke změně struktury webu či pouhému přejmenování hledaných prvků, respektive výrazů, přestane crawler pro danou univerzitu fungovat.

Konkrétní příklad, jakým způsobem je takový crawling seznamu předmětů naprogramován (zde pro univerzitu ČVUT), ukáže následující úryvek kódu:

```
...
// Projdi všechny odkazy získané na počáteční URL
for(Element e : links) {
// Pokud odkaz obsahuje tento konkrétní řetězec
    if(e.attr("abs:href").contains("predmety/00/")) {
// Přidej ho do seznamu předmětů
        linksList.add(e.attr("abs:href")); } }
...
```

Většina z univerzit má hledaný řetězec (uvedený v závorce za slovem *contains*), který je součástí odkazu, unikátní. Pouze univerzity využívající systém ECTS jsou procházeny stejně. Nicméně ČVUT, MUNI a CUNI mají získání seznamu předmětů podobně snadné, jako ilustroval úryvek kódu výše.

Systém ECTS má strukturu webu složitější, protože seznam předmětů nelze vylistovat najednou, ale postupně přes jednotlivé studijní programy. Nejprve je tedy nutné získat odkazy na tyto programy a následně se k nim opět postupně připojovat. Z odkazů získaných přes programy lze poté přejít

na studijní plány. Po připojení ke každému ze studijních plánu již získáme konkrétní odkazy na jednotlivé předměty, opět vyhledáním klíčového slova (zde *contains("plan/")*) v každém odkazu na stránce. Získané URL je poté zařazeno do seznamu odkazů (***linksList***) – tento finální krok přidání do seznamu platí pro každou z univerzit.

Podobný případ nastává i u univerzity VUTBR. Zde sice není nutno procházet jednotlivé plány či programy, avšak seznam předmětů je na webu univerzity rozdělen na „sub-stránky". Nejprve je tedy získán počet těchto stránek, na kterých se nachází jednotlivé předměty. To je provedeno vyselektováním konkrétního elementu na webu, informace je obsažena v prvním elementu , který obsahuje třídu *text*. Poté je v cyklu vytvořen seznam URL všech těchto sub-stránek, kdy je za „fixní část" URL přidáno číslo v závislosti na počtu sub-stránek. Následně se stačí připojit k jednotlivým odkazům z vytvořeného seznamu, kde dojde k vyhledání a „vyfiltrování" potřebného URL (odkazy, pro které platí – *.contains("detail-predmetu")*) všech předmětů.

Poslední „kategorií" jsou univerzity VŠE a VŠB. U nich bylo nejprve nutné odeslat formulář, aby byl získán seznam předmětů, respektive studijních plánů (u VŠE). V takovém případě je nutné znát všechny parametry, které musí být ve formuláři odeslány, aby byla jako odpověď vrácena stránka s požadovaným seznamem předmětů. Parametry odesílaného formuláře byly získány přes konzoli internetového prohlížeče (např. Mozilla Firefox), která dovolí prohlížet hlavičky odesílaných formulářů, včetně všech parametrů. Jak vypadá sestavení a odeslání formuláře pomocí knihovny *Jsoup*, ukáže následující (zkrácený) úryvek kódu:

```
Document docVSB = Jsoup.connect(startURL)
// metainformace pro weby, identifikující crawler
        .userAgent(USER_AGENT)
        .timeout(Constants.jsoupConnectionTimeout)
// vložení parametru ve tvaru „klíč:hodnota"
        .data("faculty", "5")
// odeslání formuláře metodou POST
        .post();
...
```

Po odeslání těchto formulářů již stačilo zachytit odpověď serveru a zpracovávat přijatou stránku, kde se opět dle klíčových slov prohledává seznam

odkazů na získané stránce.

Po ukončení získávání seznamu předmětů bylo ještě nutné odkazy „očistit“ o speciální znaky (prováděno již zpět v metodě *run()*). Jde o znaky, které běžně webové prohlížeče neakceptují, respektive automaticky je převádějí, což se zde neděje a je potřeba toto zařídit programově. Řeč je o znacích např. „’Á’,’Š’,’Č’“, které jsou v některých URL předmětů obsaženy. Po tomto kroce následuje už jen uložení seznamu předmětů do instance třídy univerzity (proměnná ***universities***). To se děje v *synchronizovaném* úseku kódu, aby každé vlákno mohlo do seznamu zapisovat vždy v jediný okamžik.

Crawling detailů předmětů

Pokud je vlákno třídy *SpiderLeg* vytvořeno s parametrem *false*, proběhne crawling již konkrétních informací o předmětech, na základě dříve získaného seznamu odkazů. Dojde tedy k iteraci nad kolekcí objektu univerzity (zde ***uni***, objekt získán pomocí metody ***getUnisSubjectsLinks***) obsahující odkazy na jednotlivé předměty. Pro každý z dříve uložených seznamů je zavolána metoda ***void crawlSubjects(String university, String faculty, Set<String> setOfSubjects)***. Metodě je parametrem předáno jméno univerzity, jméno fakulty dané univerzity a *Set* se seznamem URL předmětů.

V těle metody je následně vytvořen seznam (proměnná ***subjectsList***), do kterého budou ukládány jednotlivé instance třídy *Subject*, tedy objekty s informacemi o předmětu. Poté následuje rozhodnutí (větvení programu pomocí *switch*), jaké metody budou volány (na základě zkratky aktuálně procházené univerzity). Následně je již *for* cyklem procházen seznam všech URL předmětů dané univerizity a volána příšlušná „*crawlSubject...*“ metoda. Například pro univerzitu ČVUT se metoda jmenuje ***crawlSubjectCVUT***. Jejím parametrem je URL webu univerzity obsahující informace o předmětu.

Každá univerzita je tak opět procházena unikátním způsobem. Výjimku tvoří již dříve zmiňované univerzity TUL, UTB a UPCE, které využívají systému ECTS. Ten sice využívá i univerzita ZČU, avšak během crawlingu vyvstal problém, kdy bylo zjištěno, že právě ZČU má odlišně uloženou strukturu informací o předmětu.

Kvůli přehlednosti bude popis struktury webových stránek uchovávajících informace o předmětech jednotlivých univerzit rozepsán v nové kapitole, označené jako A, na straně 90, která je součástí přílohy.

Po dokončení sběru informací o předmětu je jeho objektu (***sub***) přiřazena informace o názvu univerzity (pomocí metody ***setUniversity***), názvu fakulty (metoda ***setFaculty***) a URL předmětu (metoda ***setSubject_url***). Následuje zařazení předmětu do seznamu (zmíněná proměnná *subjectsList*) a opakování procesu až do vyčerpání seznamu URL adres předmětů.

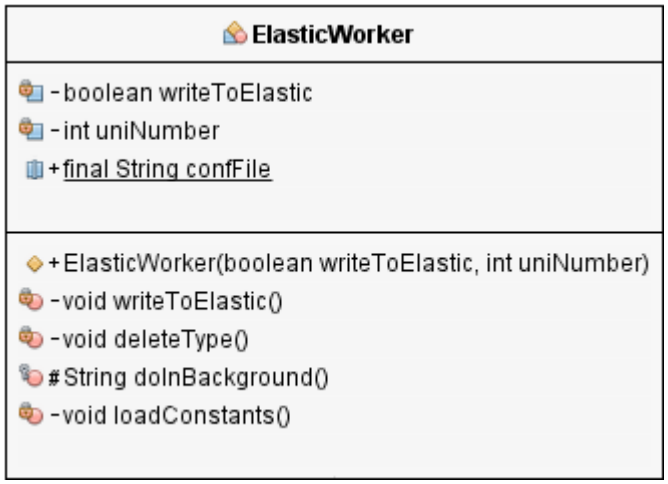
Za předpokladu, že není seznam s objekty předmětů prázdný, dochází k zápisu výsledků do požadovaných objektů, v *synchronizované* části kódu. V ní je do objektu *universities* uložena *HashMapa*, jejímž klíčem je název fakulty a hodnotou zmíněný seznam předmětů (*subjectsList*). V poli *crawlDoneUnis* je pro konkrétní číslo univerzity nastavena hodnota *true* – značí, že crawling dané univerzity proběhl bez chyby. Zároveň s tím je aktualizováno GUI crawleru, voláním metody ***updateIU***. Pomocí ní se v GUI změní ikona příslušné („ukončené“ univerzity) z červeného křížku na zelenou „fajfku“. Zároveň je aktualizován i *progress bar*. Pokud si uživatel přál uložit soubory na disk (proměnná ***saveData*** je nastavena na hodnotu *true*), dochází závěrem k volání metody ***saveFiles(String university, String faculty, ArrayList<Subject> subjectList)***.

V té je nejprve vytvořena složka v souborovém systému počítače, do které budou ukládány data. Cesta k vytvořené složce je sestavena následovně: k proměnné *saveFolder* (zadávané v GUI) je přidán znak „/“, za kterým následuje jméno univerzity (na základě metodě předané proměnné *university*), další „/“ a název fakulty (uchovává proměnná *faculty*). Následuje iterace nad seznamem předmětů (*subjectList*), kde pro každý předmět do proměnné (***fileName***) uložíme zkratku předmětu. Ta je následně „očištěna“ o speciální znaky, které jsou nahrazeny znakem „_“. Poté již dojde k samotnému uložení předmětu, do předem vytvořené složky. Výsledný soubor pro každý předmět je pojmenován jako „očištěná“ zkratka předmětu rozšířená o příponu „.json“.

8.1.5 Třída ElasticWorker

Poslední třídou cralweru je *ElasticWorker*, jejíž metody a proměnné jsou uvedeny na obrázku 8.9 níže. Při její tvorbě bylo využito převážně zdroje [9] a jeho přidružených částí. Tato třída, stejně jako *Spider*, rozšiřuje třídu *SwingWorker*, který umožňuje spustit vlákno „na pozadí“. Instance této třídy je opět založena se dvěma parametry. Prvním je *boolean* hodnota (uložená do proměnné ***writeToElastic***), která rozhodne, zda má dojít k zápisu (*true*)

či mazání dat (*false*). Druhým parametrem je číslo, které určuje konkrétní univerzitu (uloženo do proměnné ***uniNumber***).



Obrázek 8.9: Třída ElasticWorker

Nacházejí se zde dvě důležité metody. První z nich je ***writeToElastic***, která má za cíl zapsat stažená data do úložiště ElasticSearch. Tato metoda nejprve naváže spojení s databází vytvořením proměnné ***client*** typu ***TransportClient***. Následně je do této proměnné uložen ***PreBuiltTransportClient***, kterému jsou přiřazeny parametry pro připojení (adresa a port). Následuje dotaz, pomocí kterého je zjištěno, zda je úložiště dostupné. Ten vypadá následovně:

```
...
// Získání uzlů úložiště
List<DiscoveryNode> nodes = client.connectedNodes();
// Pokud je získán alespoň jeden uzel, je připojení OK
    if (!nodes.isEmpty()) { ...
```

Poté následuje několik vnořených cyklů. První z nich iteruje přes všechny univerzity (pokud je hodnota pole *crawlDoneUnis* pro příslušnou univerzitu rovna hodnotě *true*, kód pokračuje). Druhý cyklus pak iteruje přes všechny fakulty dané univerzity (aktuálně, jak bylo již zmíněno dříve, je pro každou univerzitu uložena pouze jedna fakulta). Poslední cyklus iteruje konečně přes *HashMapu* (***subMap***), ve které jsou uloženy jednotlivé předměty. Před zaindexováním do úložiště ještě dojde ke konverzi objektu předmětu do formátu JSON (pomocí knihovny *Gson*), který úložiště akceptuje. Poté je již daný

předmět uložen do databáze.

Druhou metodou je ***deleteType***, která se stará o mazání všech předmětů vybrané univerzity z úložiště. Nejprve jsou do pole řetězců (proměnná ***uniName***) zapsány konkrétní univerzity, které mají být smazány a poté proběhne připojení k úložišti (stejným způsobem, včetně kontroly dostupnosti úložiště, jako u metody pro přidání předmětů). Následně je iterováno nad zmíněným polem (*uniName*) vybraných univerzit a postupně zasílány dotazy na databázi. Pomocí nich jsou vyfiltrovány všechny odpovídající výsledky z databáze a poté již dochází k samotnému smazání dat. Konkrétní příklad jednoho z možných dotazů je následující:

```
// Vytvoření objektu dotazu
BulkIndexByScrollResponse response = DeleteByQueryAction
                    .INSTANCE.newRequestBuilder(client)
// Filtrování všech předmětů univerzity ZČU
        .filter(QueryBuilders.matchQuery("university", "ZCU"))
// Nastavení indexu databáze
        .source("universities")
// Odeslání dotazu
        .get();
```

Poslední metodou je pak ***loadConstants***, která byla krátce představena v části 8.1.3 na straně 41.

8.2 Úložiště

Zvoleným úložištěm dat je zde nástroj ***ElasticSearch***. V této sekci bude předvedeno, jakým způsobem probíhá „instalace“ a spuštění tohoto úložiště a jakým způsobem s ním lze pracovat. Tyto informace byly čerpány ze zdroje [10]. Verze úložiště použitá v této práci je 5.0.0. – ta vyšla přibližně na přelomu roku 2016 a 2017 a byla tak v práci hned použita jako novinka. Dle slov autorů přinesla oproti staré verzi (před ní byla poslední verzí 2.4.) mnoho vylepšení a novinek. Během následujících měsíců vyšlo při tvorbě této práce ještě několik nových verzí, aktuálně poslední je 5.3.2.

8.2.1 Stažení a spuštění

Nejprve je nutné celý tento nástroj stáhnout. To je možné provést zdarma na stánkách `https://www.elastic.co/downloads/elasticsearch`. Zde je k dispozici několik typů archivů, ke stažení byl zvolen hojně používaný formát *ZIP*. Jeho velikost je přibližně 32 MB (pro verzi úložiště 5.3.0.). Další postup je zde uveden pro konkrétní operační systém, Microsoft Windows 10, na kterém byla tato práce vytvořena.

Tento stažený archiv je následně nutno rozbalit na disk počítače, např. do složky *C:\ElasticSearch*. Ve zmíněné složce se po rozbalení objeví několik dalších složek a souborů. Nyní nás budou zajímat konkrétně dvě z nich, *config* a *bin*.

Spouštěcí soubory jsou uloženy ve složce *bin*. V příkazové řádce se pak stačí přepnout do pracovního adresáře této složky a spustit soubor (v případě OS Windows) *elasticsearch.bat*. Po jeho spuštění dojde k nastartování celého úložiště.

8.2.2 Nastavení

Nejprve je však vhodné úložiště nastavit. Ve složce *config* se nacházají konfugurační soubory, kterými je možno nastavovat různé vlastnosti úložiště. Nachází se v ní i soubor *elasticsearch.yml*, který je možno otevřít v libovolném textovém editoru (např. *Notepad*). Pro správnou funkci úložiště však není potřeba na základní konfiguraci nic měnit. Přesto je dobré některé pojmy z konfiguračního souboru objasnit.

Prvním z nich je název *Cluster*, který lze přeložit jako shluk. Nastavení Clusteru povoluje zvolit jeho jméno. Cluster slouží k uchování „odkazů“ na další shluky úložiště, ve kterém představuje jakýsi vrcholek hierarchie. Pod něj pak spadají *Nodes*, neboli uzly.

Uzel představuje fakticky jednu z mnoha možných stanic Clusteru, ke kterému je připojen. Díky tomu je tuto databázi možno rozložit na několik různých strojů „hlásících“ se do stejného Clusteru. To se hodí v případě, kdy přestává stačit výkon serveru – jednuduše se přidá nový uzel do stejného Clusteru a Elastic data efektivně rozdělí s ohledem na nově přidaný server. V případě této práce je však uzel pouze jediný, proto ve výchozím nastavení

není potřeba opět nic měnit.

Další užitečnou položkou může být *Path*, kde lze zadat adresu v počítači, kam mají být ukládána data, respektive logy úložiště. Zde opět ponecháno ve výchozím nastavení. Data jsou ukládána do složky *data*, logy ve složce *logs* – obě jsou součástí adresáře, do kterého bylo úložiště rozbaleno.

Na závěr ještě zmínka k nastavení připojení k úložišti. V části s názvem *Network* lze nastavit IP adresu a port, pod kterým bude daná stanice, respektive uzel, dostupný pro ostatní členy Clusteru. Nastavení v části *Discovery* pak slouží pro řízení komunikace mezi uzly Clusteru. Ve výchozím nastavení probíhá komunikace s úložištěm přes adresu *127.0.0.1* a port *9300*. Na stejné adrese, ale portu *9200*, pak lze s úložištěm komunikovat pomocí protokolu *HTTP* (Hypertext Transfer Protocol).

8.2.3 Komunikace s úložištěm

Po případném nastavení úložiště je nutné ho spustit. Jak již bylo zmíněno dříve, stačí spustit soubor *elasticsearch.bat* ze složky *bin*. Poté již nebrání nic tomu začít s úložištěm komunikovat. Jako *index* pro uložení dokumenů bylo zvoleno slovo *universities*, pod které spadají jednotlivé *typy* – zkratky univerzit. Toto členění lépe objasní příklady uvedené v textu níže.

cURL

Prvním možným způsobem, jak s úložištěm pracovat, je pomocí příkazů přes nástroj *cURL*. Jedná se o nástroj příkazové řádky, který slouží k přenosu dat a webovou komunikaci. Ten je nutné dodatečně nainstalovat, není totiž součástí OS Windows. Nejedná se však o příliš uživatelsky přívětivou variantu. Následující příklad představí příkaz GET, ve kterém bude získán konkrétní předmět ze ZČU s určitým (zde smyšleným, z důvodu zkrácení zápisu) ID:

```
curl -XGET 'localhost:9200/universities/ZCU/AVtdErk28?pretty'
```

V uvedeném příkladu je nejprve adresa a port, na kterých úložiště naslouchá HTTP požadavkům. Za číslem portu následuje lomeno a index, pod

kterým jsou data uložena. Druhé lomítko odlišuje typ dokumentu, zde tedy dokumenty pro ZČU, za kterým následuje už samotný index. Dodatečnou možností je pak „?pretty“, která zajistí „hezké“ zobrazení získaného dokumentu (není povinné).

Kibana

Druhým způsobem je využití grafického doplňku k úložišti ElasticSearch s názvem *Kibana*. Tento nástroj je opět možné zdarma stáhnout ze stránek Elastic na adrese `https://www.elastic.co/downloads/kibana`. „Instalace“ je obdobná, jako bylo zmíněno v sekci 8.2.1 výše. Spuštění proběhne zadáním příkazu *kibana.bat*. Nejprve je však nutné mít spuštěno samotné úložiště ElasticSearch. V nastavení pro Kibanu je pouze nutné dbát na to, aby hodnoty adresy a portu korespondovaly s nastavením těchto hodnot pro ElasticSearch (aby bylo možné se k němu správně připojit). Po spuštění už stačí v libovolném internetovém prohlížeči zadat adresu „localhostu“ s portem *5601* (tedy konkrétně např. *http://localhost:5601/)*.

Po zadání zmíněné adresy se otevře grafické rozhraní (viz obrázek 5.1 na straně 19), ve kterém nás bude aktuálně zajímat položka levého menu s názvem *Dev Tools*. Po jejím rozkliknutí se objeví část rozdělená do dvou panelů, kde v levém z nich lze zadávat příkazy, pravý zobrazuje odpovědi úložiště.

Jako příklad lze opět uvést vyhledání dokumentu (podobně jako za použití výše zmíněného cURL), který v tomto prostředí vypadá následovně (zde již s existujícím celým ID):

```
GET /universities/ZCU/AVtdEr6W8v6Fa6lV2A49
```

V tomto případě není nutno ani uvádět možnost „?pretty“. Po odeslání dotazu je na pravé straně obrazovky zobrazen vyhledaný dokument.

Důležitým krokem je pak nastavení úložiště na vyhledávání v českém jazyce. Pro přehlednost je konkrétní nastavení uloženo jako snímek obrazovky s příkazem. Tento snímek zobrazuje obrázek 8.10 níže. Informace k nastavení byly čerpány ze zdrojů [6] a [7].

Stopwords v tomto případě značí slova, která mají být vyjmuta z inde-

```
PUT /universities/_settings
{
  "settings": {
    "analysis": {
      "filter": {
        "czech_stop": {
          "type":       "stop",
          "stopwords":  "_czech_"
        },
        "czech_stemmer": {
          "type":       "stemmer",
          "language":   "czech"
        }
      },
      "analyzer": {
        "czech": {
          "tokenizer":  "standard",
          "filter": [
            "lowercase",
            "czech_stop",
            "czech_stemmer"
          ]
        } } } } }
```

Obrázek 8.10: Ukázka nastavení českého jazyka pro úložiště

xování a vyhledávání. V češtině se jedná například o slova „a“, „ale“, „či“, „tak“ atd. Stemmer se pak stará o zpracování jednotlivých slov při jejich indexaci – např. rozklad slova na jeho základní tvar.

Aby bylo možné toto nastavení nad určitým indexem provést, je nejprve nutné index „uzavřít“ pomocí příkazu

```
POST /universities/_close
```

a následně provést příkaz uvedený na obrázku (8.10). Poté je možné index opět „otevřít“, pomocí následujícího příkazu:

```
POST /universities/_open
```

Mezi dalšími příkazy v prostředí Kibana pak lze uvést *_search*, který slouží

pro vyhledávání dokumentů v databázi. Následují dva příklady použití vyhledávání dokumentů. První uvedený vyhledá všechny dokumenty pod indexem *universities* a zobrazí jeho přesný počet dokumentů. Druhý pak udělá obdobný dotaz, který je však konkretizován pro *typ* ZČU.

```
// Vyhledání všech dokumentů
GET /universities/_search
// Vyhledání dokumentů konkrétního typu
GET /universities/ZCU/_search
```

Následující úryvek kódu pak ukazuje část výsledku pro druhý dotaz, ve kterém se dozvíme přesný počet uložených dokumentů pro ZČU:

```
"hits": {
// Počet dokumentů
    "total": 790,
// Nehledáme konkrétní dokument, score 1 pro všechny
    "max_score": 1,
// Následuje výčet dokumentů
    "hits": [ ...
```

Vyhledávat lze pak i pomocí tzv. *query* výrazů, kdy hledáme určitý text v konkrétních polích dokumentu, v celém indexu, nezávisle na zvoleném typu. Příkladem lze uvést vyhledání spojení „počítačové sítě“ v polích „name“ a „annotation“, který vypadá následovně:

```
GET /universities/_search {
// Chceme pouze 2 výsledky, od prvního nalezeného
    "from": 0, "size": 2,
    "query": {
// Multimatch = hledáme ve více polích najednou
        "multi_match": {
            "query":"počítačové sítě",
            "fields": [ "name", "annotation"]
} } }
```

Implicitní nastavení pro vyhledávání víceslovných dotazů je takové, že v každém z hledaných polí je dotaz rozdělen na jednotlivá slova a vyhledávání

probíhá za pomocí klauzule *OR*. Tedy pro výraz „počítačové sítě“ jsou v polích „name“ a „annotation“ vyhledávána slova „počítačové“ *NEBO* „sítě“ (mohou být i obě zároveň). Vyhledávání lze případně nastavit i na klauzuli *AND*. V takovém případě musí prohledávané pole obsahovat všechny výrazy z víceslovného dotazu [20].

Nastavení těchto klauzlí je nutné provést přímo při hledání výrazu, pomocí zavolání vlastnosti *operator*. V případě Java API je pak nutný zásah do kódu programu, aby bylo toto změněno. Nastavení je následující:

```
QueryBuilders.matchQuery(field, value)
// změníme vyhledávání dotazu na výraz AND
            .operator(MatchQueryBuilder.Operator.AND);
```

Vhodné je dále jmenovat příkaz *POST*, pomocí kterého lze vkládat dokumenty do úložiště. Příklad pro vložení dokumentu do úložiště pod index *universities* a typ ZCU je následující:

```
POST /universities/ZCU/ {
    "faculty": "FAV", "shortcut": "OOP", "credits": 6 }
```

Poslední jmenovaný bude příkaz *DELETE* podobný příkazu *POST*. Pomocí něj dokážeme smazat dokumenty z úložiště. To jde navíc kombinovat s příkazem pro vyhledání konkrétních dat, což je v crawleru využito pro mazání všech dat konkrétní univerzity. Od verze ElasticSearch 5.0.0. a vyšší nelze využít příkaz, pomocí kterého bylo možno z úložiště smazat všechny dokumenty konkrétního typu. Příkaz pro smazání dat tak může, s využitím *query*, vypadat následovně:

```
POST universities/ZCU/_delete_by_query {
    "query": {
        "match": {
// Smaž všechny předměty z katedry KIV
            "department": "KIV"} } }
```

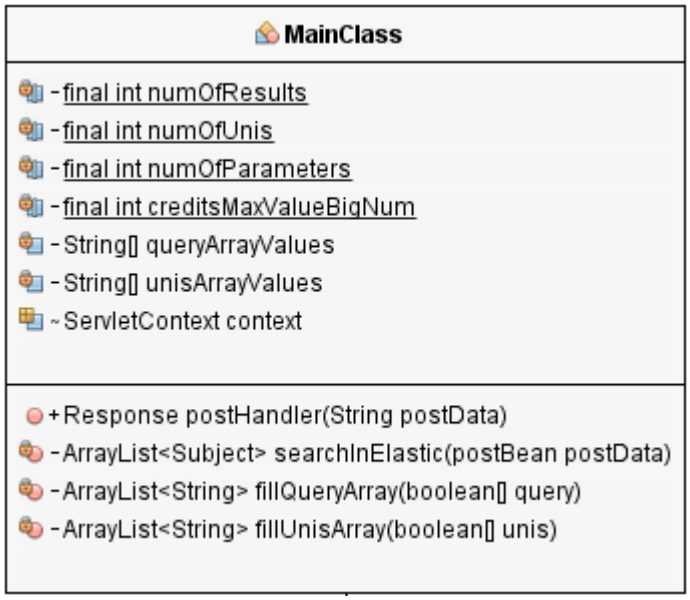
V programové (Java) části této práce pak probíhá komunikace s úložištěm přes knihovny poskytované tvůrci ElasticSearch. Ty obsahují API se všemi potřebnými metodami pro přidávání, výběr a mazání dat.

8.3 Server

Serverová část práce je naprogramována pro provoz na serveru *Apache Tomcat* (verze 7) s *Java EE 6*. Ke správě *buildu* aplikace bylo využito nástroje *Apache Maven*, projekt tedy obsahuje soubory pro něj specifické. Mezi ty patří například soubor *pom.xml*, který obsahuje jak metainformace k samotnému sestavení projektu, tak *dependencies*, které určují, jaké knihovny budou k projektu přiřazeny. Mezi nimi lze pak jmenovat *elasticsearch.client* sloužící pro práci s úložištěm, *jersey.core* pro nastavení serveru a servletů, *sun.jersey* pro práci s dokumenty formátu JSON či *code.gson*, opět sloužící pro zpracování JSON dokumentů.

8.3.1 Hlavní část

Server se sestává z jedné hlavní třídy, ***MainClass*** (její metody a funkce ukazuje obrázek 8.11 níže), a dvou dalších tříd, které slouží pouze pro vytvoření objektů předmětu (třída ***Subject***, která je shodná se stejně pojmenovanou třídou z crawleru, popsanou v sekci 8.1.1 na straně 34, respektive obrázku 8.4 v oné sekci) a POST požadavku (třída ***postBean*** – viz obr. 8.12 na straně 58). Komunikace s webovou částí serveru pak probíhá přes rozhraní *REST*.



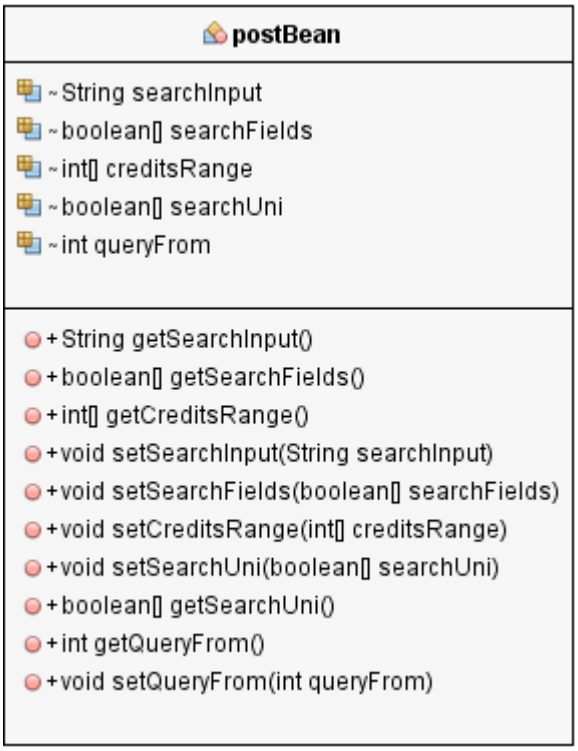
Obrázek 8.11: Třída MainClass

Nejprve je tedy nutné „namapovat“ třídu *MainClass* a její metody na konkrétní adresu, na které budou „naslouchat“ pro příchozí spojení, respektive požadavky. Více o *RESTu* viz [25].

Toto mapování je díky použitým knihovnám (*jersey*) naprosto jednoduché. Přes anotaci *@Path* uvedenou před jménem třídy určíme, pod jakým mapováním bude daná třída na serveru „naslouchat“. Následující úryvek kódu toto objasní prakticky:

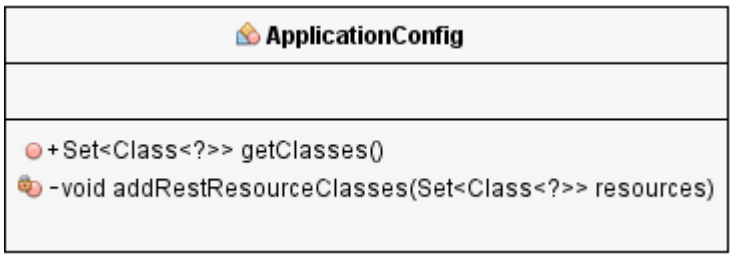
```
@Path(”elasticServer”)
public class MainClass { ...
```

Zmíněná třída *postBean* pak obsahuje proměnné odpovídající JSON dokumentu odesílanému z JavaScriptové části serveru. Více ke struktuře tohoto dokumentu viz sekce 8.4.2 na straně 71, konkrétně v části věnující se funkci pro komunikaci se serverem.



Obrázek 8.12: Třída postBean

Poslední třídou je pak ***ApplicationConfig***, která slouží pouze k nastavení výsledné aplikace. Tato třída vznikla při založení webového projektu ve vývojovém prostředí a neměla by být ručně modifikována. Její vlastnosti si lze prohlédnou na obrázku 8.13 níže. V její metodě ***addRestResourceClasses*** je jako *resource* nastavena třída *MainClass*.



Obrázek 8.13: Třída ApplicationConfig

8.3.2 Metody třídy MainClass

Hlavní metodou této třídy je ***public Response postHandler(String postData)***. Ta zpracovává požadavky zasílané uživateli z webové části aplikace. Opět zde probíhá určité mapování, doplněné o několik dodatečných informací. Metoda je mapována na název *serverPost*, je typu *POST* a zároveň produkuje a přijímá dokumenty ve formátu JSON. Nastavení v kódu pak vypadá následovně:

```
@Path("serverPost") // Mapování cesty na název „serverPost“
@POST // nastavení typu metody (POST či GET)
@Consumes(MediaType.APPLICATION_JSON) // Přijímá JSON
@Produces(MediaType.APPLICATION_JSON) // Odesílá JSON
public Response postHandler(String postData) throws IOException {
...
```

Tato metoda tedy vezme příchozí JSON dokument (proměnná ***postData***), převede jej na objekt třídy *postBean* (kvůli pozdějšímu zpracování) a vyvolává metodu ***ArrayList<Subject> searchInElastic(postBean postData)***. Výsledek této metody je poté uložen do proměnné ***reply*** typu *ArrayList<Subject>*. Na závěr metody *postHandler* je zpět uživatelskému rozhraní odeslán návratový kód s případnými daty. Návratová hodnota vypadá následovně:

```
...
// Odešli data proměnné reply s návratovou hodnotou 200
return Response.status(200).entity(reply).build();
...
```

Dříve zmíněná metoda ***searchInElastic*** převezme data přijatá z webové části (strukturu dat, která server přijímá, si lze prohlédnout v sekci 8.4.2, na straně 73) a postupně je zpracuje. Nejprve dojde k nastavení rozsahu kreditového ohodnocení pro hledaný předmět. Při volbě hodnoty „6+“, tedy „maxima“ kreditů, bylo nutno pomoci si nastavením maximální hodnoty pro kredity. Toto číslo je uloženo v konstantní proměnné ***creditsMaxValueBigNum*** a je rovno 100, což jednoznačně obsáhne všechny předměty. Hodnoty pro maximum a minimum kreditů jsou uloženy do proměnných ***creditsMin***, respektive ***creditsMax***. Text hledaného výrazu je uložen do proměnné ***query***.

Následuje volání další metody, ***ArrayList<String> fillQueryArray (boolean[] query)***. Té je předáno pole *boolean* hodnot, které určují „oblast vyhledávání“. Výstupem této metody je *ArrayList* obsahující zmíněné oblasti. Ten je na základě pole *query* naplněn hodnotami *name*, *course_outline* a *annotation* – vždy minimálně jednou z nich.

Poté je do proměnné ***unisArray*** přiřazen další *list*, získaný metodou ***ArrayList<String > fillUnisArray(boolean[] unis)***. Ta provádí velmi podobný kód jako dříve zmíněná metoda *fillQueryArray*. V tomto případě však přiřazuje zkratky vybraných univerzit (opět na základě pole *boolean* hodnot předaného této metodě).

Poté dojde k připojení k úložišti (stejně jako v případě crawleru, pro více detailů viz sekce 8.1.5 na straně 48) a následují tzv. *QueryBuilders*. Pomocí nich dojde k sestavení dotazu odesílaného na databázi. Tyto *Query* dotazy jsou vytvářeny způsobem *multiMatchQueries*, který umožní zadání více prohledávaných položek v jednom dotazu. Díky tomu lze sestavit dotaz, kdy je uživatelem zadaný výraz vyhledáván ve všech třech nabízených položkách. Úryvek této části kódu vypadá následovně:

```
// query = hledaný výraz
// searchArray obsahuje „name“, „course_outline“ či „annotation“
MultiMatchQueryBuilder mmqb = QueryBuilders.multiMatchQuery(
      query, searchArray.toArray(new String[searchArray.size()])); ...
```

Tento dotaz je poté předán jako parametr pro ***searchRequestBuilder***, který je již odesílán na databázi. Dále je tomuto dotazu ještě nutné nastavit *Types*, tedy mezi kterými univerzitami bude dotaz vyhledáván (informace dříve uložená v proměnné *unisArray*). Další nastavení, *PostFilter*, zajistí, aby byly obsaženy výsledky pouze v požadovaném rozsahu kreditního ohodnocení předmětů. *Filter* vypadá následovně:

```
...
// zpracováváme položku „credits“
.setPostFilter(QueryBuilders.rangeQuery(”credits”)
        .from(creditsMin) // rozsah OD
        .to(creditsMax) // rozsah DO
        .includeLower(true) // včetně hodnoty OD
        .includeUpper(true)) // včetně hodnoty DO
...
```

Na závěr se ještě nastaví „stránkování“, tedy kolik výsledků a s jakým „posunem“ mají být vráceny, což ukáže následující kód:

```
...
// numOfResults je konstantní proměnná nastavená na hodnotu 10
.setFrom(postData.queryFrom).setSize(numOfResults);
...
```

Nutné je ještě zmínit, že musí být explicitně uveden *SearchType*, který je nastaven na hodnotu *QUERY_THEN_FETCH*. To znamená, že dotaz nejprve provede požadované operace nad databází. Až poté vrátí přesné výsledky, odpovídající požadovaným hodnotám. Odeslání dotazu a uložení jeho výsledků vypadá následovně:

```
...
// odeslání dotazu a uložení odpovědi
SearchResponse response = searchRequestBuilder
                                .execute().actionGet();
// získání dokumentů z odpovědi serveru
SearchHit[] results = response.getHits().getHits();
...
```

Na závěr již následuje jen iterování nad výsledky dotazu, kdy je každý dokument formátu JSON převeden na objekt předmětu (pomocí proměnné

typu *ObjectMapper*) a přidán do pole výsledků. Zároveň je mu přiřazeno i tzv. *HitScore*, které uživateli přiblíží, který z vyhledaných předmětů přesněji odpovídá dotazu. Toto skóre je přiřazováno automaticky úložištěm a značí shodu jeho záznamu s hledaným výrazem. Standardní algoritmus, používaný v *ElasticSearch*, počítá skóre na základě tří následujících položek (zdroje [13] a [14]):

- **Term frequency**
  uvádá četnost výskytu hledaného slova v dokumentu – čím častější výskyt, tím vyšší skóre.
- **Inverse document frequency**
  značí, jak často se hledaný výraz vyskytuje ve všech dokumentech pro daný index, ve kterém prohledáváme (v této práci index *universities*).
- **Field-length norm**
  je vypočtena na základě délky pole, ve kterém je vyhledáváno. Pokud tedy hledáme výraz „programování v jazyce C“ v poli „anotace“, je vrácená hodnota pro tento výraz tím nižší, čím delší je celý obsah pole „anotace“ konkrétního dokumentu.

8.4 Webová aplikace

Tato část práce slouží pro zobrazení informací koncovému uživateli. Jedná se o webovou stránku, která by měla být volně přístupná uživatelům na internetu. Právě přes ní bude probíhat dotazování na server a zobrazování výsledků těchto dotazů. Na základě výsledků pak bude uživatel moci porovnat jednotlivé předměty (v rámci jejich popisů a kreditního ohodnocení). Jednotlivé předměty si pak bude moci označovat za „zajímavé“. Na základě tohoto výběru mu pak bude doporučena univerzita vhodná ke studiu.

Webová aplikace je rozdělena na část s HTML kódem, který je podpořen CSS styly a „výkonnou“ částí vytvořenou v JavaScriptu. Jednotlivé soubory budou podrobněji popsány v následujících částech.

Při tvorbě webové části této práce byl kladen důraz na co nejjednodušší design a snadno pochopitelné ovládání. Web tak neobsahuje příliš mnoho rozmanitých barev. Vlastně je navržen pouze v kombinaci modré barvy s bílou, s černými texty. Přesto je design dle mého názoru moderní a přehledný.

8.4.1 HTML část

Část pro zobrazení obsahu webu se skládá z celkem tří souborů. Jedná se konkrétně o dokumenty *index.html*, *mainPage.html* a *favSubjects.html*. Obsah těchto souborů bude popsán v následujících částech textu. Podrobný výpis kódu či struktury zde uveden kvůli přehlednosti nebude. Ostatně, zdrojový kód HTML stránek si může prohlédnout každý návštěvník výsledného webu. Metody a proměnné zmíněné v této části textu jsou provázány s JavaScriptovou částí, blíže popsanou v sekci 8.4.2 na straně 71.

index.html

Vstupním bodem pro HTML část je zde soubor *index.html*. Ten je uživateli zobrazován při zadání URL adresy serveru v internetovém prohlížeči. Hned v počátečním tagu (<*html*>) je uvedena *direktiva Angularu* (*ng-app*), která „propojuje“ celou stránku s JavaScriptovým modulem (tedy propojení zobrazovací HTML části s funkční částí uloženou v souboru *myJS.js*).V hlavičce (<*head*>) jsou pak připojeny všechny nezbytné soubory pro zobrazení webu, podrobněji popsané níže. Dále se v ní nachází metainformace o webové stránce, kterými jsou *description* pro popis webu, *keywords* pro určení klíčových slov na webu, *author* se jménem autora, *viewport* pro definici rozlišení stránky a nakonec *charset*, určující znakovou sadu pro zobrazení obsahu (nastaveno na univerzální kódování *UTF-8*).

Ze vzdálených serverů jsou stahovány dodatečné skripty a CSS styly. Mezi ty patří soubory *bootstrap.min.css* a *bootstrap.min.js* obsahující styly a JavaScriptový kód, nezbytné pro správnou funkčnost Bootstrapu. Soubor se skripty Jquery (*jquery.min.js*), opět nezbytný pro správnou funkčnost Bootstrapu. Další vzdáleně stahované soubory patří Angularu (*angular.js* a *angular-animate.js*), bez kterých by opět nebylo možné Angular korektně používat. Rovněž je připojen i soubor umožňující zobrazení a použití „slideru“ určeného pro výběr rozsahu kreditů (soubor *rzslider.js*).

Lokálně jsou pak připojeny 3 CSS a jeden soubor s JS (JavaScript) kódem. CSS soubory slouží pro „stylování“ samotného designu webu (*mainCSS.css*), pro styl animace zobrazené při čekání na odpověď serveru (*loadingAnimationCSS.css*) a poslední pro styl posuvníku (*sliderCSS.css*), pomocí kterého lze volit počet kreditů. Soubor s JS (*myJS.js*) pak obsahuje kód pro komunikaci se serverem a podporu pro zobrazování HTML stránky.

Tělo dokumentu (<*body*>) se skládá z tagů <*div*>, pomocí kterých je nejprve vytvořeno záhlaví stránky. Zároveň zde přichází propojení s kontrolerem JS části, což zajišťuje direktiva *ng-controller*. Poté dochází k „připojení“ dalšího obsahu (zbylých dvou HTML stránek) pomocí direktivy *ng-include* na základě proměnné ***includePage***. Ta značí, zda má být zobrazen obsah úvodní stránky (hodnota 1 – *mainPage*, zobrazena při prvním načtení webu) či stránky se seznamem oblíbených předmětů (hodnota 2 – *favSubjects*).

Jak webová stránka vypadá, po jejím prvním načtení, ukazuje obrázek 8.14 níže. Jedná se o výřez horní části webu, zbývající spodní část je po prvním načtení pouze bílá plocha.

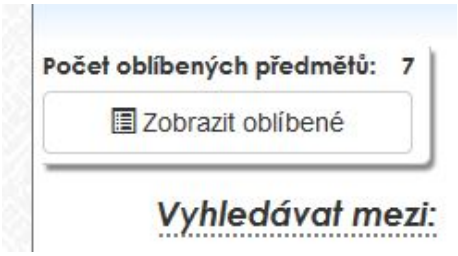
Obrázek 8.14: Výřez z webu po jeho načtení

mainPage.html

Tato HTML část slouží pro zobrazení ovládacích prvků, pomocí kterých může uživatel provést vyhledávání doporučených předmětů.

V levé horní části této stránky se nachází „počitadlo oblíbených předmětů“, znázorněné na obrázku 8.15 níže. To udává počet předmětů, které uživatel označil za zajímavé. Toto zobrazení je opět „obaleno“ do několika tagů <*div*> pro správné pozicování elementu. Počet zvolených předmětů je zobrazen v závislosti na proměnné ***favSubsCount*** z JS části webu. V případě, že je mezi „oblíbenými“ vybrán alespoň jeden předmět, je na základě direktivy *ng-show* zobrazeno tlačítko, které slouží pro přepnutí se na stránku se seznamem vybraných předmětů. Samotné přepnutí pak probíhá prostou změnou proměnné ***includePage*** po kliknutí na tlačítko (zajistí direktiva *ng-click*).

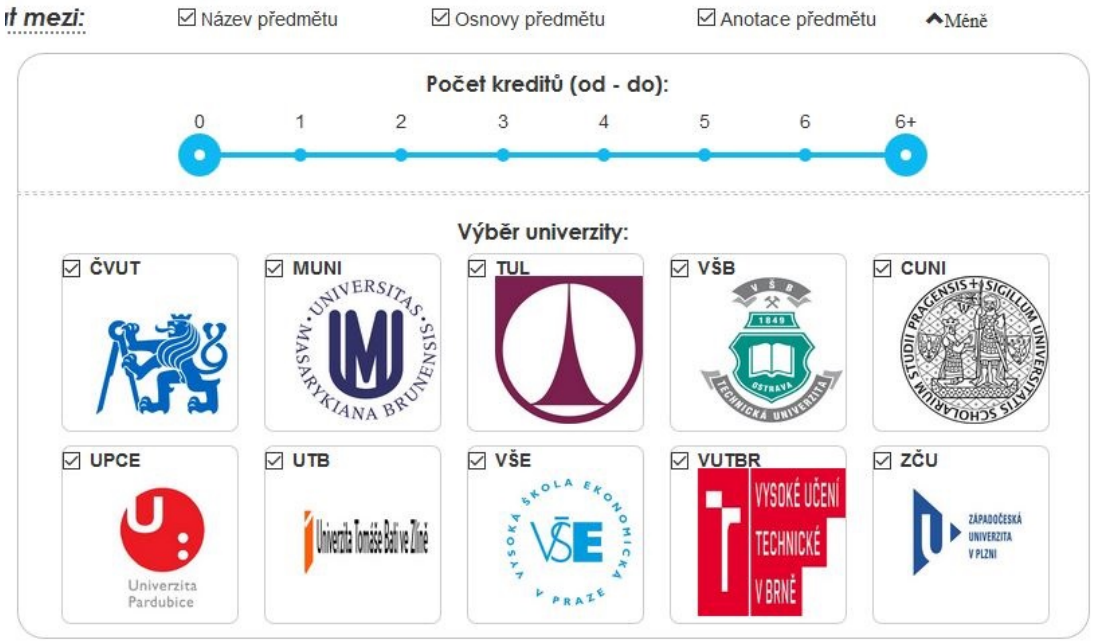
V prostřední sekci webu se nachází oblast s *checkboxy* (zaškrtávací tlačítka), pomocí kterých lze navolit, ve které části dat bude položený dotaz vyhledáván. Na výběr jsou celkem tři možnosti: *název*, *osnovy* a *anotace* předmětu. Při



Obrázek 8.15: Část webu ukazující počet oblíbených předmětů

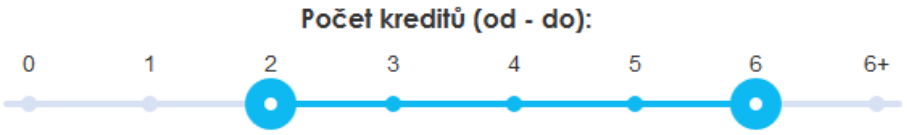
odesílání dotazu na databázi musí být vybrána alespoň jedna z těchto tří možností (implicitně jsou zvoleny všechny). Jednotlivé checkboxy jsou opět pomocí direktivy Angularu (*ng-model*) vázány na konkrétní proměnnou – zde pole tří možností s hodnotami *true* či *false*, jehož název je ***searchCheck***.

Vedle těchto polí se nachází text, který po kliknutí zobrazí, respektive skryje, více možností pro vyhledávání. Hodnoty „zobraz/skryj“ jsou přepínány pomocí direktivy *ng-if*. Kliknutí na tento text zároveň vyvolá metodu ***moreOptions***, díky které dojde k překreslení posuvníku pro výběr počtu kreditů. To je nutné pro jeho správné zobrazení při otevření „panelu“ více možností. Jak vypadá část se zobrazenými možnostmi lze vidět na obrázku 8.16 níže.



Obrázek 8.16: Zobrazené dodatečné možnosti

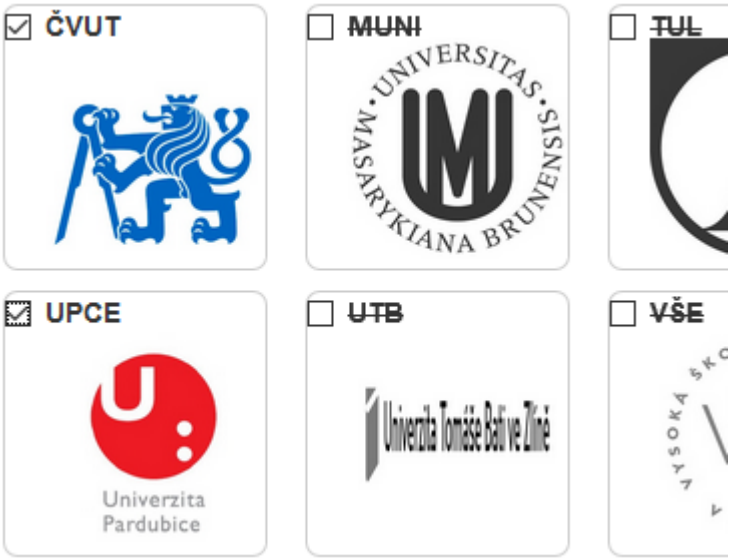
Pod volbou více možností se pak nachází samotná část s ovládacími prvky pro specifičtější výběr předmětů. Celá tato oblast „spadá“ pod <*div*> obsahující atribut třídy ***moreOptionBody***. Zobrazení probíhá na základě proměnné ***moreSearchOptions***. Nejprve je v této části vykreslen posuvník určený pro výběr počtu kreditů pro vyhledávané předměty. Posuvník je sestaven pomocí speciálního tagu *rzslider* a je provázán s proměnnou ***slider***. V té lze nastavit minimální a maximální hodnoty posuvníku. Jako minimum je zde hodnota 0 – některé předměty mají i nulové kreditní ohodnocení. „Maximum“ je zde označeno jako *6+*, tedy 6 a více kreditů (například některé předměty ke státním závěrečným zkouškám mohou mít ohodnocení i 15 kreditů). Posuvník tedy uživateli umožňuje nastavit rozsah, respektive minimální a maximální hodnoty pro kreditní ohodnocení vyhledávaných předmětů. Implicitně je zvolen rozsah 0 až „maximum“ (tedy 6+). Jak je tento posuvník vyobrazen na webu ukáže obrázek 8.17.



Obrázek 8.17: Posuvník pro výběr počtu kreditů

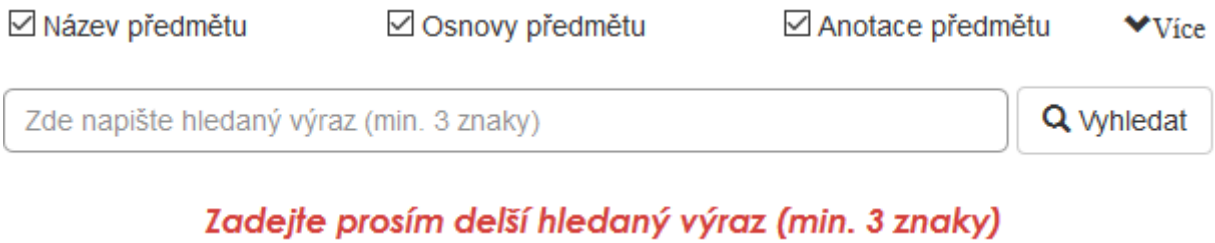
Pod posuvníkem následuje výpis seznamu univerzit (aktuálně obsažených v této práci) situovaný do dvou řádků pod sebe. Tento seznam je opět zhotoven jako checkboxy pro každou z univerzit. K těmto prvkům je dále přiřazen text zkratky příslušné univerzity a zmenšený obrázek jejího loga. Výběr univerzit je opět zhotoven pomocí direktivy *ng-model* a propojení s příslušným polem hodnot s názvem ***searchUni***. Implicitně jsou vybrány všechny univerzity. Pokud si uživatel nepřeje vyhledávat předměty na některé z nich, stačí klepnutím na logo dané univerzity zrušit výběr checkboxu. Logo se poté stane černobílým a zkratka univerzity se přeskrtne, aby byly lépe vizuálně odděleny univerzity, které nejsou vybrány. Tuto část si lze jako výřez z webové stránky prohlédnou na obrázku 8.18 o stránku níže.

Pod celým tímto výběrem je zobrazeno pole, které slouží pro zadávání textu dotazů uživatele. Vpravo od něj se nachází tlačítko pro odeslání dotazu. Jeho modelem je proměnná ***inputScope***. Zmíněné tlačítko po stisknutí vyvolá metodu ***postClick***, která obstará komunikaci se serverem.



Obrázek 8.18: Výřez z výběru univerzit

Pokud dojde během komunikace k chybě, je tato chyba vypsána pod zmíněným polem pro hledaný výraz. Chyby jsou vypsány zvětšeným červeným písmem. Obrázek 8.19, který ukazuje výpis chyby, kdy se uživatel pokusil vyhledat výraz kratší než 3 znaky, je zobrazen níže.



Obrázek 8.19: Výpis chybového hlášení uživateli

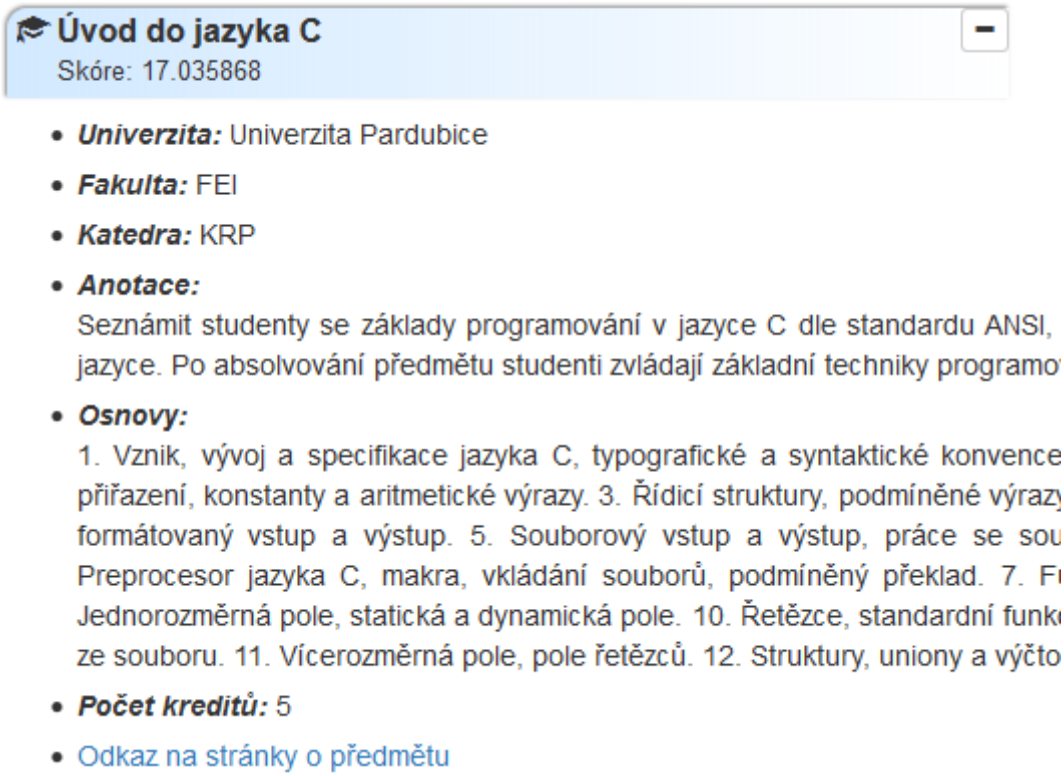
Pod vstupním polem se rovněž zobrazuje „načítací kolečko“, které je zobrazeno pomocí direktivy *ng-show* na základě proměnné ***showLoader***. Ta je nastavena na hodnotu *true*, pokud si uživatel vyžádá nová data ze serveru. *False* je přiřazeno až poté, co uživatel obdrží odpověď ze serveru – ať již s výsledky dotazu či nějaké z chybových hlášení. Zmíněný „loader“ je vyhotoven jako animace, pomocí poměrně složitého CSS kódu, a složen je z několika tagů <*div*>. Ukázka *loaderu* je na obrázku 8.20 na straně 68.

Obrázek 8.20: Čekání na odpověď

Po odeslání dotazu jsou uživateli zobrazeny výsledky vyhledávání (design výsledků viz obrázek 8.21 níže – výsledky na obrázku odpovídají hledanému výrazu „programování v jazyce C“). Ty jsou na webu situovány do levé části a zobrazeny jen v případě, že výsledkem dotazu je alespoň jeden předmět z databáze. Opět se jedná o soustavu vnořených tagů <*div*>. Získaná data jsou iterována pomocí direktivy *ng-repeat*, která prochází celé pole výsledků (proměnná ***data***) a zobrazí jednotlivé předměty z tohoto pole. U každého předmětu je implicitně zobrazena „hlavička“ s názvem předmětu a tzv. *hitScore*. To udává shodu hledaného výrazu s položkami v databázi – vyšší skóre značí lepší shodu (pro více detailů viz závěr sekce 8.3.2 na straně 59.
Vpravo od této „hlavičky“ se pak nachází tlačítko označené jako „+“, pomocí kterého lze přidat příslušný předmět mezi „oblíbené“. Kliknutí na toto tlačítko vyvolává metodu ***addSubToInterested*** pro přidání, respektive ***removeSubFromInterested*** pro odebrání předmětu.

Obrázek 8.21: Zobrazení výsledků vyhledávání

„Hlavička“ je zhotovena jako klikatelný prvek – kliknutí má za následek zobrazení více informací o předmětu. Tyto informace jsou zobrazeny pomocí nečíslovaného seznamu (<*ul*>). Jak zobrazené detaily vypadají, znázorňuje obrázek 8.22.



Obrázek 8.22: Detaily předmětu po jeho rozkliknutí

Na konci seznamu zobrazených předmětů se nachází tlačítko, pomocí kterého lze načíst dalších 10 výsledků se stejnými vyhledávacími kritérii, jako měl poslední hledaný výraz. Tlačítko je kvůli přehlednosti omezeno na 5 kliknutí, lze tedy načíst dodatečných 50 záznamů k 10 původně zobrazeným. Zobrazení tlačítka ukazuje obrázek 8.23 uvedený níže.

Obrázek 8.23: Tlačítko pro načtení dalších výsledků

favSubjects.html

Tato HTML stránka je strukturou velmi podobná zmíněné *mainPage.html*. Dvě níže jmenované části mají téměř totožný HTML kód. V horním levém rohu webu se opět nachází „počitadlo oblíbených předmětů“ a pod ním tlačítko, které v tomto případě navrací uživatele zpět na část webu s vyhledáváním předmětů.

Na levé straně obrazovky se rovněž nachází seznam předmětů, tentokrát těch přidaných do seznamu „oblíbených“. Opět je nejprve zobrazena jejich „hlavička“ (zde již bez skóre), kterou lze rozkliknout pro detailnější popis předmětu. Tlačítko se znakem „-“ na pravé straně „hlavičky“ odebere předmět z tohoto seznamu. Tím je předmět nenávratně odstraněn z tohoto výpisu – přidat ho zpět lze pouze jeho opětovným vyhledáním v databázi.

V pravé horní části obrazovky se pak nachází výčet univerzit odpovídající vybraným předmětům. Přesněji řečeno výčet, kolik těchto předmětů spadá pod jetnotlivé univerzity. Toho je opět docíleno pomocí direktivy *ng-repeat* nad objektem ***unisMetainfo***. V tom je pod hodnotou ***interestedSubs*** uloženo číslo, které udává počet vybraných předmětů pro konkrétní univerzitu. Na tento objekt je ještě aplikován filtr *orderObjectBy*, pomocí kterého jsou řazeny univerzity sestupně, dle počtu uživatelem vybraných předmětů. Ukázku použití filtru znázorňuje následující úryvek kódu:

```
...
<div ng-repeat=”university in
unisMetainfo | orderObjectBy:’interestedSubs’:true”>
...
```

Poslední část této stránky se nachází vpravo, pod zmíněným výčtem univerzit. Jedná se o doporučenou univerzitu. Ta je zvolena na základě počtu vybraných předmětů pro jednotlivé univerzity. Pokud některá z nich obsahuje více „oblíbených“ předmětů než ostatní, stává se z ní univerzita pro uživatele doporučená. V této části je pak zobrazeno její logo. V případě že žádná z nich nemá vyšší počet vybraných předmětů než ostatní, je zde zobrazena pouze hláška „Na základě aktuálního výběru nelze rozhodnout“. Jak tato část vypadá v praxi, na základě náhodného výběru několika předmětů, znázorňuje obrázek 8.24 níže.

Obrázek 8.24: Doporučená univerzita

8.4.2 JavaScriptová část

JavaScriptová podpora webu je postavena na technologii *AngularJS* vyvíjené společností Google.

Základním prvkem je pak definice *module*. Do libovolně pojmenované proměnné (zde ***app***) je vložena instance modulu. Definice modulu obsahuje jeho jméno a několik (pro tuto práci) nutných doplňků, uvedených v hranatých závorkách. Vytvoření modulu vypadá následovně:

```
var app = angular.module('ServerModule', ['rzModule',
                                'ngAnimate', 'ui.bootstrap']);
...
```

ServerModule je v tomto případě název modulu. Jednotlivé doplňky jsou pak *rzModule*, pomocí kterého je vykreslován, respektive zpracováván, posuvník pro výběr počtu kreditů. Doplněk *ngAnimate* pak slouží například pro direktivu *ng-hide*, která skrývá elementy na základě „vázané“ proměnné. Modul *ui.bootstrap* je pak nutný pro zpracování prvků doplňku *Bootstrap*.

K vytvořenému modulu je následně přiřazen *controller* starající se o komunikaci s HTML částí. Kontroler má podobnou syntaxi jako zmíněný modul, tedy následující:

```
...
app.controller('MyCtrl', function($http, $scope, $timeout) {
...
```

„Navázán“ je na dříve definovaný modul, respektive proměnnou ***app***. Výčet parametrů, uvedených v závorce za slovem *function*, udává, jaké typy proměnných budou v kontroleru použity. Zde *$http* pro odesílání a přijímání http komunikace s Java serverem, *$scope* pro využití „globálních“ proměnných v kontroleru a poslední *$timeout*, který zde obstarává překreslení posuvníku při jeho skrytí či zobrazení.

Aby bylo možno Angular správně využívat, je nutné, jak modul, tak kontroler, „provázat“ ve správné hierarchické struktuře s HTML tagy. Přesněji řečeno musí být nejprve přiřazen modul (pokud jich web neobsahuje více, bývá obvykle přiřazen „nejvyššímu“ tagu *<html>*) a poté, vnořený v něm, kontroler. Těch může aplikace využívat i více (v této práci však postačil jediný). Toto přiřazení HTML tagům je popsáno i v části 8.4.1 na straně 63.

Zmíněný kontroler tedy v této práci obstarává „vše“ podstatné v rámci JavaScriptu. Nejprve definuje globální (*$scope*) proměnnou ***slider***, která slouží pro účely zmíněného posuvníku, kterým lze nastavit počet kreditů. V této proměnné se určuje minimální a maximální hodnota posuvníku.

Na několika následujících řádcích jsou v tomto JS dokumentu definovány globální proměnné nezbytné pro správný chod webu. Mezi ně patří například ***unisMetainfo***. Jedná se o asociativní pole, které jako klíč obsahuje název (respektive zkratku – např. „ZCU“) univerzity a hodnotou je pak JSON pole s dodatečnými informacemi pro jednotlivé univerzity. Těmito informacemi jsou celý název univerzity (položka ***name***), zkratka (včetně háčků a čárek – položka ***shortcut***), název loga univerzity (***logo***) uloženého ve složce *images* na serveru a počet oblíbených předmětů na dané univerzitě (***interestedSubs***).

V této části kódu se pak nachází ještě první funkce, ***moreOptions***, která se stará o překreslení posuvníku a nastavení hodnoty globální proměnné ***moreSearchOptions*** na *true* či *false* – v závislosti na tom je zobrazeno či skryto více možností pro výběr univerzity.

Funkce pro komunikaci se serverem

Dále pak následuje několik dalších funkcí. První z nich, která je zároveň tou „nejdůležitější“, je ***postClick***. Ta je vyvolána po kliknutí na tlačítko pro vyhledávání výrazu v databázi. V této funkci nejprve proběhne kontrola – zda uživatel zadal do vyhledávacího pole alespoň 3 znaky, zda je vybrána alespoň jedna ze tří „oblastí“ pro vyhledávání (*název* předmětu, *osnovy* či*anotace*) a zda-li je zvolena alespoň jedna z univerzit. V případě, že některá z těchto podmínek není splněna, je do globální proměnné ***checkError*** vložena příslušná chybová hláška, která je poté zobrazena uživateli. V případě, že žádná chyba nenastala, pokračuje kód dále. Dojde k nastavení proměnné ***clickMoreNum*** na hodnotu 0, což značí, že dosud nebylo zmáčknuto tlačítko pro načtení dodatečných výsledků hledání. Následně je zobrazen „loader“, nastavením proměnné ***showLoader*** na hodnotu *true*. Poté je sestaven JSON dokument, který obsahuje data odesílaná na server. Jeho formát je následující:

```
$scope.postJSON = {
        ”searchInput”:$scope.inputScope,
        ”searchFields”:$scope.searchCheck,
        ”creditsRange”:[$scope.slider.minValue,$scope.slider.maxValue],
        ”searchUni”:$scope.searchUni,
        ”queryFrom”:0
    };
```

Proměnná ***searchInput*** obsahuje vyhledávaný text, pole ***searchFields*** pak „oblast vyhledávání“ (název předmětu, osnovy či anotace – pro každou z těchto položek má hodnotu *true* či *false*). Pole dvou hodnot ***creditsRange*** pak obsahuje minimální a maximální hodnoty nastavené v posuvníku, ***searchUni*** je opět pole hodnot pravda/nepravda obsahující informaci, které z univerzit budou obsaženy ve výsledcích hledání. Poslední položka, ***queryFrom***, udává „posun“ pro vyhledávané výsledky. Nejprve je posun nastaven na 0. Jelikož databáze vrací pouze 10 výsledků pro hledaný výraz, umožňuje příslušné tlačítko zvětšit tento posun, což vede k načtení nových záznamů z databáze.

Následuje pak využití proměnné *$http*. Pomocí ní je sestaven a odeslán dotaz na databázi. Je uvedena metoda (zde *POST*), URL, kam je dotaz zasílán (POST je mapován na URI */DPElasticServer/rest/elasticServer/serverPost*),

data obsahující dříve zmíněný JSON dokument a hlavička dotazu udávající, že bude odesílán JSON dokument.
Následují dvě funkce, ***successCallback*** a ***errorCallback***. První z nich je vyvolána v případě, že návratová hodnota odpovědi ze serveru je „vpořádku". V této metodě pak dojde k přiřazení přijatých dat do globální proměnné ***data*** a nastavení proměnné ***searchDone*** udávající, že vyhledávání již doběhlo. Závěrem je proměnná *showLoader* nastavena na *false*, čímž je skryt loader.
Druhá metoda je vyvolána v případě neúspěchu, ať již kontaktování serveru, či negativní odpovědi serveru (například při nemožnosti spojit se s databází).

Obdobnou činnost, jako funkce *postClick*, pak zastává další z funkcí – ***moreResultsClick***. Jak naznačuje její název, je vyvolána po kliknutí na tlačítko, které slouží pro načtení dalších záznamů z databáze (pro stejně položený dotaz utvořený dříve, při vyvolání funkce *postClick*). Rozdíl je takový, že již nekontroluje počet zadaných znaků do vyhledávacího pole, či jaké položky pro vyhledávání jsou „zaškrtnuty". Nejprve zvětšuje globální proměnnou ***clickMoreNum***, která je následně ověřována, zda je menší než 6. Pokud ano, je v JSON poli (proměnná ***postJSON***), které je odesíláno na server, navýšena hodnota ***queryFrom*** o 10. Poté proběhne odeslání dotazu a jeho zpracování, stejně jako v případě první jmenované metody. V případě kladné odpovědi od serveru pouze přidá nová data k již existujícím, místo nahrazení dosavadních dat novými.

Ostatní funkce

Další funkcí je pak ***addSubToInterested***. Té je předán objekt konkrétního předmětu. Funkce nejprve na základě dat objektu vytvoří hash (vyvolá funkci ***subjectHash*** popsanou níže). Pokud se hash ještě nenachází v globálním poli ***interestedSubsArray***, je do něj přidán (pod klíčem vytvořeného hashe) jako JSON struktura. Ta obsahuje samotný předmět a hodnotu ***visible***, pomocí které lze na webu skrývat či zobrazovat detaily o předmětu. Následně funkce zvýší počet oblíbených předmětu o 1, stejně jako zvýší hodnotu ***interestedSubs*** v poli ***unisMetainfo*** o 1 pro univerzitu, pod kterou předaný předmět spadá. Na závěr ještě vyvolá funkci ***setMostFavoriteUni***, která zajistí výběr „doporučené" univerzity.

Funkce ***removeSubFromInterested*** má pak opačné chování než funkce popsaná v odstavci výše. Jejím cílem je odebrat daný předmět z „oblíbených" a zároveň snížit všechny odpovídající hodnoty o 1.

Následující funkce, ***isInInterested***, na základě vypočteného hashe zkontroluje, zda se předmět nachází mezi oblíbenými. Pokud ne, vrací hodnotu *false*, pokud ano, tak hodnotu *true*.

Dříve zmíněná funkce ***subjectHash***, které je parametrem předán objekt předmětu, vytvoří na jeho základě textový hash. Aby byl tento hash pro každý předmět unikátní, je vytvořen složením názvu univerzity, fakultou předmětu, katedrou předmětu a zkratkou předmětu. V hashi jsou poté odstraněny nežádoucí vzniklé znaky a je metodou navrácen zpět volajícímu. Vytvoření hashe ukazuje následující kód:

```
$scope.subjectHash = function(subject) {
var subHash = subject.university+subject.faculty+
              subject.department+subject.shortcut;
// odstranění nechtěných znaků
subHash = subHash.replace(/\s+/g, ''); ...
```

Poslední metodou je pak již zmíněná ***setMostFavoriteUni***. Ta prochází pole ***unisMetainfo***, ve kterém kontroluje počty vybraných předmětů jednotlivých univerzit a v cyklu tak vybere tu, která obsahuje více předmětů než všechny ostatní. Pokud se nějaká taková najde, je do proměnné ***mostFavoriteUni*** tato univerzita přiřazena. Pokud ne, je stejná proměnná nastavena na hodnotu *false*.

Posledním úsekem kódu je filtr, přiřazený jedinému modulu tohoto JavaScriptového souboru. Ten je přiřazen podobně jako dříve zmíněný kontroler. Hlavička filtru vypadá následovně:

```
app.filter('orderObjectBy', function() {
...
```

V těle funkce pak probíhá řazení asociativního pole, pro které je daný filtr použit. Výstupem je seřazené pole, na základě požadované proměnné.

8.4.3 CSS část

Webová část této práce využívá celkem 3 lokální CSS soubory. Dva z nich byly staženy z volně dostupných zdrojů na internetu a následně použity

v projektu pro zobrazení „loaderu“ a posuvníku pro výběr počtu kreditů. Jedná se o soubory ***loadingAnimationCSS.css*** a ***sliderCSS.css***. Jejich obsah nebude podrobněji rozebírán, protože, jak bylo již řečeno, se jedná o volně stažitelné úryvky kódu, jejichž funkcionalitu není nutné znát, důležité je pouze správné využití těchto stylů.

mainCSS.css

Tento soubor již obsahuje vlastní originální kód. Veškeré stylování textů a grafiky na výsledném webu je obsaženo právě v něm.

Nachází se zde mnoho selektorů tříd, mezi kterými lze jmenovat například ***.heading*** pro zobrazení hlavičky či ***.heading h1*** pro stylování nadpisu v hlavičce. Tagům *<html>* a *<body>* je zde pak přiřazen obrázek textury na pozadí.

Pomocí těchto selektorů je stránka dělena na „sekce“. Tedy levý sloupeček s výsledky vyhledávání, prostřední sloupeček s nastavením vyhledávání, pravá část pak na stránce s vybranými předměty obsahuje doporučenou univerzitu atd.

Jelikož je v této práci použita technologie CSS verze 3, jsou využity některé její nabízené možnosti. Mezi ty patří například přechody při zobrazování, respektive skrývání, prvků. To je realizováno pomocí funkce *transition*. Ta je v různých internetových prohlížečích interpretována rozdílně, proto je nutné v CSS souboru uvést definici pro všechny možné případy. Následující kód je ukázkou definice této funkce pro nejpoužívanější prohlížeče:

```
...
-webkit-transition: background 0.5s; /* pro Safari */
   -moz-transition: background 0.5s; /* pro Firefox */
     -o-transition: background 0.5s; /* pro Operu */
        transition: background 0.5s; /* standardní syntaxe */
...
```

Podobně bylo nutné definovat i lineární přechody barvy pozadí vybraných prvků (například hlavička celého webu či hlavička vyhledaných předmětů). Standardní syntaxe pro lineární přechod barvy (uvedené v RGBA zápise, tedy včetně průhlednosti) vypadá následovně:

```
...
background: linear-gradient(to right, rgba(166,212,255,0.5) 0%,
                                      rgba(255,255,255,1) 100%);
...
```

Mezi další „moderní“ prvky webu lze pak řadit zaoblené rámečky okolo prvků, které lze definovat jednoduchou vlastností prvku, konkrétně například ***border-radius: 7px;***. Lepšímu vzhledu pak napomáhá i stínování prvků webu, které přidává optické výraznosti a slouží k lepšímu odlišení jednotlivých prvků. Stín lze prvku přidat vlastností selektoru např. ***box-shadow: 2px 3px 2px #888888;***, kde první tři čísla (určené v pixelech) udávají velikost a posun stínu, čtvrté číslo pak udává jeho barvu.

Ostatní vlastnosti selektorů pak patří již k těm „základním“, mezi kterými lze jmenovat např. *padding* a *margin* sloužící pro odsazení prvků, práce s textem pomocí *font-size* či *font-family*, způsob zobrazení prvku pomocí vlastnosti *display*, určení statických rozměrů pomocí *width* a *height* či v neposlední řadě vlastnost *float* pro zarovnání prvku na požadovanou pozici.

Podrobný popis jednotlivých selektorů zde uveden nebude, jedná se totiž o poměrně rozsáhlý soubor, který je pro čtenáře stejně nepodstatný – v případě zájmu si lze selektory prohlédnout ve zmiňovaném CSS souboru. Dalším dobrým způsobem, jak „proniknout“ do konkrétního nastavení selektorů v této práci je využití konzole internetového prohlížeče. V té si pak lze jednoduše vybrat požadovaný prvek webové stránky a uživateli je zobrazeno, jaké všechny selektory a vlastnosti tento prvek obsahuje. Tato užitečná funkce byla hojně využíváná i při „ladění“ designu výsledného webu.

9 Testování a možná rozšíření

V této kapitole bude popsán způsob testování výsledné práce, jeho výsledky a úpravy aplikace na základě připomínek uživatelů, kteří systém testovali. V závěrečné sekci pak bude práce zhodnocena a navrženy další možné úpravy a změny v aplikaci.

9.1 Testování uživateli

Aplikace byla poskytnuta minimálně pěti různým uživatelům pro testování výsledné práce. Představeny jim byly aplikace crawleru a výsledná webová část práce. Cílem testů bylo zjistit, zda uživatel, který se s touto prací setkal poprvé, bude schopen bez problému výsledné aplikace používat.

9.1.1 Crawler

Na testování této části byl ve výsledku kladen menší důraz než na webovou aplikaci, ke které budou ve finále přistupovat zájemci o studium na vysoké škole. Aplikace crawleru bude sloužit pouze pro administrátora systému, proto nebylo ve výsledku nutno řešit každý drobný detail či nedostatek.

První připomínkou, kterou uživatelé testující práci sdíleli, byl fakt, že crawler žádným způsobem nezobrazoval progres své vykonané práce. Na základě těchto připomínek byl do aplikace dodělán *progress bar*, který alespoň částečně dává uživateli představu, kolik „práce“ je již hotovo a kolik zbývá.

Další výtka směřovala k ukládání stažených dat na disk. Problém je ten, že stažená data z disku jsou pouze uložena a není s nimi dále manipulováno, nahrávání dat do databáze je možné pouze přes crawler, který musí nejprve stáhnout aktuální data. Tento „problém“ nebyl dále řešen. Mohlo by docházet k tomu, pokud by crawler umožňoval nahrávat data uložená na disku do databáze, že stažená data nemusí být aktuální a koncový uživatel by dostával nepravdivé informace. Druhou nevýhodou by pak bylo dvojí uchovávání stejných dat. Přesto se tato funkce může hodit pro případnou archivaci dat, protože úložiště by mělo obsahovat každý rok aktualizovaná

data s tím, že stará data neuchovává.

Připomínky byly rovněž vzneseny k mazání dat z databáze. To zprvu probíhalo okamžitě po stisknutí tlačítka pro smazání dat. Toto provedení bylo špatné z toho důvodu, že uživatel mohl omylem kliknout na příslušné tlačítko a nechtěně data smazat. Proto bylo k události stisku tlačítka přidáno vyskakovací okno, které se uživatele dotáže, zda si opravdu přeje smazat vybraná data.

Jednou z připomínek bylo taktéž to, že „dosažitelnost“ úložiště není kontrolována před samotný spuštěním crawleru. Systém je nyní nastaven tak, že nejprve je nutné provést crawling dat a až poté je možné data uložit. Pokud však databáze není spuštěna, dozví se toto uživatel až po skončení crawlingu, když se pokusí stažená data uložit. Tento problém již nebyl v rámci práce vyřešen, nejedná se o zásadní věc – crawling (v aktuální podobě práce) běží v průměru 3 až 4 minuty. Pokud tedy nebude databáze dostupná a uživatel se toto dozví až po stažení dat, není ztracený čas nijak zásadně velký, avšak do budoucna by bylo vhodné toto ošetřit.

Další z námitek pak byla technického rázu. Údaje potřebné pro připojení k úložišti byly v kódu crawleru napevno uloženy. To však nebylo vhodné v případě nasazování práce na server, pokud by bylo nutné měnit implicitní hodnoty (IP adresa a port, na kterém je úložiště spuštěno). Proto byl vytvořen JSON soubor obsahující tyto informace – ten je nutné mít uložený ve stejné složce jako program crawleru. Pokud je tento konfigurační soubor nepřítomen, budou brány implicitní hodnoty, které jsou stále v kódu crawleru nastaveny.

9.1.2 Webová aplikace

Testování této části je naopak důležité. S webovou aplikací přijde do kontaktu cílová skupina uživatelů a proto je nutné, aby web dobře vypadal a zároveň fungoval bez problémů.

Pozitivní na testování webové aplikace byl fakt, že i uživatelé, kteří nepatří k těm zkušenějším na práci s počítači, se na webu snadno zorientovali a dokázali vyhledat informace a přidávat si předměty do seznamu „oblíbených“.

V první řadě si uživatelé pochvalovali velmi jednoduchý a jasný design, ke

kterému bylo jen pár drobných výtek, snadno opravených úpravou CSS stylů. Mezi hlavní zápory pak patřila výtka, že hledaný výraz nelze odeslat pouhým stisknutím tlačítka *Enter* na klávesnici. To bylo záhy opraveno přídáním formuláře, do kterého bylo vloženo vyhledávací pole.

Další drobností je pak připomínka, že pokud dotaz vrátí méně než 10 předmětů, je zobrazeno tlačítko pro načtení dalších. Jeho stisknutí v tomto případě pouze vypíše, že nelze nalézt další předměty (ovšem po stisknutí tlačítko již správně zmizí). To je logické, protože již první dotaz nezobrazí ani požadované minimum. Jedná se o drobný detail, které neni do buducna obtížné opravit.

Některým uživatelům pak přišlo tlačítko pro přidávání předmětů mezi „oblíbené“ nepraktické, ve formě pouhého tlačítka se znakem plus. Po najetí myší nad toho tlačítko je však zobrazena nápověda, k čemu tlačítko slouží. To je dle mého názoru dostačující řešení.

9.2 Zhodnocení a návrhy na rozšíření

Na základě nejčetnějších výtek od uživatelů, kteří aplikaci testovali, byla práce upravena do finální podoby, která již bude prezentována koncovým uživatelům. Cíle této práce se podařilo naplnit, ačkoliv systém v současné podobě poskytuje uživatelům možnost vyhledávat pouze mezi informatickými předměty na omezeném počtu vysokých škol.

První rozšíření práce je tedy nasnadě – upravení crawleru, aby bylo možné stahovat data z více vysokých škol. To vyžaduje nastudování struktury webů dalších univerzit a následný zásah do kódu crawleru, kde bude nuté upravit kód obstarávající jak získání seznamu všech předmětů, tak vytvoření nových metod, které se postarají o stažení podrobných dat o jednotlivých předmětech. Odhadem by přidání nové univerzity (respektive její jedné fakulty) mohlo zabrat přibližně 5 až 6 hodin práce – tedy pokud je dotyčný konající úpravy seznámen s problematikou crawlingu informací. Na základě existujícího kódu lze však tuto činnost poměrně rychle pochopit. Nároky kladené na úložný prostor v počítači nejsou momentálně příliš veliké, uložená data zabírají přibližně 9 MB. V případě rozšiřování práce však lze očekávat, že tyto nároky nebudou přímo úměrně zvětšovány s narůstajícím objemem dat v úložišti.

Momentálně je kód crawleru připraven na stahování předmětů i z ostatních kateder vybraných deseti univerzit v této práci. Aktuálně by mělo stačit do seznamu kateder přidat jméno „nové“ katedry a její počáteční URL se seznamem předmětů – tedy za předpokladu, že i ostatní katedry daných univerzit dodržují stejnou hierarchii a strukturu uložení předmětu, jako aktuálně procházené katedry.

S přidáním nových univerzit či kateder však vyvstane nutnost upravit GUI crawleru i webové aplikace. Aktuálně je toto vytvořeno „na míru“ 10 univerzitám. V rámci crawleru je tedy nutné vyřešit způsob, jakým si administrátor bude volit, pro kterou z univerzit chce crawling spustit. V případě webové části by rovněž bylo nutno upravit seznam univerzit, mezi kterými si uživatel vybírá, zda se mají vyskytovat ve výsledcích hledání. To samé pak platí i při pouhém přidání více kateder k jednotlivým univerzitám – v tomto případě je však jedním z řešení ponechat možnost pouze na výběru celých univerzit a zobrazovat či stahovat vždy všechny jejich katedry najednou.

Dalším návrhem na budoucí rozšíření tohoto projektu je přidání předmětů vyučovaných v anglickém jazyce. Aktuálně jsou (pokud to výběr předmětů umožňuje) filtrovány pouze předměty vyučované v češtině. S tím rovněž souvisí nastavení databáze. Ta je momentálně nastavena na vyhledávání v českém jazyce, což znevýhodňuje předměty s anglickým popisem jejich osnov a anotací. Řešením by bylo uchovávat data v úložišti pod dvěma různými indexy, kdy jeden by byl nastaven na vyhledávání v jazyce českém a druhý v jazyce anglickém. Zvolené úložiště tuto možnost poskytuje.

S předchozí možností rovněž souvisí přidání zahraničních univerzit. Opět by bylo nutné, jako v případě přidání dalších českých univerzit, vybrat a nastudovat strukturu webů a upravit příslušné části práce pro tuto možnost. Zde však vyvstává otázka, na kolik by vynaložená práce byla užitečná, kolik procent studentů středních škol plánuje pokračovat ve studiu na zahraniční vysoké škole.

Vhodným rozšířením by byla také úprava doporučení univerzity na základě vybraných předmětů. Aktuálně se tak děje pouze na základě počtu vybraných předmětů pro dané univerizty, vhodné by však bylo zařadit k tomuto i sumu skóre vybraných předmětů, výsledek by poté byl relevantnější, uživateli by byla spíše nabídnuta univerzita, která odpovídala hledaným frázím.

10 Závěr

V rámci diplomové práce byly prozkoumány webové stránky vybraných univerzit a na základě získaných znalostí byl naprogramován webový crawler. Pomocí něj je možné stáhnout požadovaná data o předmětech vyučovaných na těchto univerzitách. Dalším krokem pak byl výběr vhodné databáze, pro uchování získaných dat, která zároveň zvládne rychlé fulltextové vyhledávání v uložených dokumentech. Finálním krokem pak bylo navržení a tvorba webových stránek, které uživatelům poslouží pro snadné vyhledávání dat ze zmíněné databáze.

Čtenář se v textu práce dozví, jaké možnosti jsou dnes v oblasti web crawlingu dostupné, stejně tak jako jaký je vhodný typ úložiště pro ukládání získaných dat. Dále je mu také představena struktura webových stránek vybraných univerzit. V kapitole popisující praktickou část práce se dozví, jakým způsobem jsou sestrojeny a provázány jednotlivé komponenty této práce, což může být užitečné i pro uživatele, kteří si chtějí postavit vlastní crawler. Závěrem textu se lze dočíst několik návrhů na možná vylepšení stávajícího systému.

Účelem této práce je všem zájemcům o studiu na vysoké škole poskytnout unikátní možnosti ve vyhledávání mezi předměty vyučovanými na vybraných vysokých školách. Cílovou skupinou jsou tedy žáci středních škol posledních dvou ročníků, kteří ještě nejsou pevně rozhodnuti, na kterou z univerzit podat přihlášku pro své další studia. Právě díky této diplomové práci se na základě výsledků vyhledávání v databázi předmětů mohou snáze rozhodnout pro konkrétní vysokou školu.

Aktuálně je práce zaměřena pouze na informatické předměty vyučované na deseti vybraných univerzitách. O výsledek této práce je však zájem i mezi dalšími českými univerzitami, proto by bylo vhodné tuto práci dále rozšířit. Zásadním krokem je zvětšení jejího rozsahu na více univerzit a vyučovaných oborů. Momentálně tato práce slouží spíše jako demonstrativní příklad, jakým způsobem je možné získávat a interpretovat data z webů univerzit.

Zároveň může tato práce posloužit i jako výchozí studie pro rozsáhlejší grantovou aktivitu. Tu by vzhledem k povaze práce a proklamovaným prioritám měly státní orgány podpořit. Režie spojená s údržbou tohoto systému je pak minimální, k provozu postačí veřejně přístupný server, všechna potřebná data spojená s touto prací zabírají aktuálně méně než 200 MB. Zároveň by

univerzity ve vlastním zájmu dodaly administrátorovi tohoto systému své studijní programy, respektive vyučované předměty, v požadovaném tvaru, což by opět snížilo časové náklady pro rozšiřování a údržbu této práce.

Použité zkratky

API – Application Programming Interface
BSON – Binary JavaScript Object Notation
CSS – Cascading Style Sheets
csv – Comma-separated values
DB – Databáze
ECTS – European Credit Transfer System
EE – Enterprise Edition
GUI – Graphic User Interface
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
IP – Internet Protocol
JDK – Java Developement Kit
JS – JavaScript
JSON – JavaScript Object Notation
MB – Mega Byte
OS – Operační systém
REST – Representational State Transfer
RSS – Rich Site Summary
SE – Standard Edition
SGML – Standard Generalized Markup Language
SQL – Structured Query Language
URL – Uniform Resource Locator
UTF – Unicode Transformation Format
W3C – World Wide Web Consortium
XML – Extensible Markup Language

Literatura

[1] *AngularJS: A Powerful JavaScript Framework* [online]. [cit. 22.4.2017]. Dostupné z: https://www.upwork.com/hiring/development/angularjs-basics/.

[2] *Bootstrap* [online]. [cit. 21.4.2017]. Dostupné z: http://whatis.techtarget.com/definition/bootstrap.

[3] *Co je XML?* [online]. [cit. 17.4.2017]. Dostupné z: https://www.interval.cz/clanky/co-je-xml/.

[4] *Study of Web Crawler and its Different Types* [online]. [cit. 12.4.2017]. Dostupné z: http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue1/Version-6/A016160105.pdf.

[5] *Úvod do CSS* [online]. [cit. 21.4.2017]. Dostupné z: http://www.webtvorba.cz/css/uvod-do-css.html.

[6] *Elasticsearch: Vyhledáváme hezky česky* [online]. [cit. 29.4.2017]. Dostupné z: https://www.zdrojak.cz/clanky/elasticsearch-vyhledavame-cesky/.

[7] *Language analyzers* [online]. [cit. 29.4.2017]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-lang-analyzer.html.

[8] *What is Elasticsearch, and How Can I Use It?* [online]. [cit. 19.4.2017]. Dostupné z: https://qbox.io/blog/what-is-elasticsearch.

[9] *Java API* [online]. [cit. 27.4.2017]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/client/java-api/5.0/index.html.

[10] *Setup Elasticsearch* [online]. [cit. 28.4.2017]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/5.0/setup.html.

[11] *MongoDB vs. Elasticsearch: The Quest of the Holy Performances* [online]. [cit. 19.4.2017]. Dostupné z: http://blog.quarkslab.com/mongodb-vs-elasticsearch-the-quest-of-the-holy-performances.html.

[12] *Naučte se pracovat s jedním z nejúspěšnějších prodejních kanálů na internetu aneb jak funguje Heureka.cz* [online]. [cit. 24.3.2017]. Dostupné z: https://ebrana.cz/blog/naucte-se-pracovat-s-jednim-z-nejuspesnejsich-prodejnich-kanalu-na-internetu-aneb-jak-funguje-heurekacz-3-cast.

[13] *What is Relevance?* [online]. [cit. 6.5.2017]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html.

[14] *How scoring works in Elasticsearch* [online]. [cit. 6.5.2017]. Dostupné z: https://www.compose.com/articles/how-scoring-works-in-elasticsearch/.

[15] *How a Web Crawler Works: Insights into a Modern Web Crawler* [online]. [cit. 12.4.2017]. Dostupné z: https://www.promptcloud.com/how-web-crawler-works/.

[16] *Seznámení s HTML 5* [online]. [cit. 21.4.2017]. Dostupné z: https://www.interval.cz/clanky/seznameni-s-html-5/.

[17] *What is HTML?* [online]. [cit. 21.4.2017]. Dostupné z: http://www.yourhtmlsource.com/starthere/whatishtml.html.

[18] *Parsování HTML v Javě s knihovnou Jsoup* [online]. [cit. 24.4.2017]. Dostupné z: https://www.itnetwork.cz/java/pokrocile/java-knihovna-jsoup-parsovani-html.

[19] *Jsoup Tutorial and Examples* [online]. [cit. 25.4.2017]. Dostupné z: http://howtodoinjava.com/jsoup/complete-jsoup-tutorial/.

[20] *Multi Match Query* [online]. [cit. 9.5.2017]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html.

[21] *Nabídkové agregátory* [online]. [cit. 7.4.2017]. Dostupné z: https://www.jaknainternet.cz/page/1650/nabidkove-agregatory/.

[22] *NoSQL (Not Only SQL database)* [online]. [cit. 18.4.2017]. Dostupné z: http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL.

[23] *Search Engine with Apache Nutch, MongoDB and Elasticsearch* [online]. [cit. 14.4.2017]. Dostupné z: http://www.aossama.com/search-engine-with-apache-nutch-mongodb-and-elasticsearch/.

[24] *Portia vs. ParseHub comparison* [online]. [cit. 14.4.2017]. Dostupné z: https://blog.parsehub.com/portia-vs-parsehub-comparison-which-alternative-is-the-best-option-for-web-scraping/.

[25] *REST: architektura pro webové API* [online]. [cit. 30.4.2017]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/.

[26] *Začínáme blogovat: pomohou RSS agregátory!* [online]. [cit. 9.4.2017]. Dostupné z: http://michalkubicek.cz/zaciname-blogovat-rss-agragatory/.

[27] *3 Examples of Parsing HTML File in Java using Jsoup* [online]. [cit. 16.4.2017]. Dostupné z: https://www.javacodegeeks.com/2014/09/3-examples-of-parsing-html-file-in-java-using-jsoup.html.

[28] *Úvod do JSON* [online]. [cit. 17.4.2017]. Dostupné z: http://www.json.org/json-cz.html.

[29] *Web Crawler* [online]. [cit. 12.4.2017]. Dostupné z: https://www.techopedia.com/definition/10008/web-crawler.

[30] *Naučte se pracovat s jedním z nejúspěšnějších prodejních kanálů na internetu jak funguje Zbozi.cz* [online]. [cit. 24.3.2017]. Dostupné z: https://ebrana.cz/blog/naucte-se-pracovat-s-jednim-z-nejuspesnejsich-prodejnich-kanalu-na-internetu-aneb-jak-funguje-zbozicz-2-cast.

[31] BING, L. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.* : Springer Berlin Heidelberg, 2007. ISBN 978-3-540-37882-2.

[32] GORMLEY, C. – TONG, Z. *Elasticsearch: The Definitive Guide.* : O'Reilly Media, 2015. ISBN 978-1449358549.

[33] HILLS, T. *NoSQL and SQL Data Modeling.* : Technics Publications, 2016. ISBN 978-1634621090.

[34] MCCREARY, D. G. – KELLY, A. M. *Making Sense of NoSQL.* : Manning Publications, 2013. ISBN 978-1617291074.

[35] MITCHELL, R. *Instant Web Scraping with Java.* : Packt Publishing, 2013. ISBN 978-1849696883.

Seznam příloh

A Struktura webů s informacemi o předmětu

Tato kapitola podrobněji popíše strukturu webových stránek konkrétních předmětů vybraných univerzit. Uveden bude vždy jeden předmět reprezentující každou z univerzit, na kterém bude popsáno, jak vypadá jeho struktura HTML stránek a jakým způsobem jsou jednotlivé stránky parsovány.

Text této kapitoly je nedílnou součásti popisu crawleru ze sekce 8.1.4 na straně 43, která byla kvůli přehlednosti vyčleněna zvlášť. Následuje tedy výčet vybraných univerzit s detailnějším popisem struktury stránek jejich předmětů.

A.1 České vysoké učení technické v Praze

Pro popis struktury byl jako příklad náhodně vybrán předmět *Internet a klasifikační metody*, jehož detaily se nacházejí na `http://bk.fit.cvut.cz/cz/predmety/00/00/00/00/00/00/02/96/12/p2961206.html`.

Nejprve je v crawleru zjišťováno, zda stránka vůbec obsahuje anotace k předmětu – při procházení bylo nalezeno několik předmětů, které tuto informaci neobsahovaly a zároveň neobsahovaly ani žádné dodatečné informace o předmětu, kromě jména. Proto je vyhledán nejprve tento element.

Struktura webu je pak tvořena několika tabulkami. První z nich obsahuje základní informace o předmětu, kterými jsou jméno předmětu, jeho zkratka, kreditové ohodnocení atd.

Nejprve je vyselektována zkratka předmětu, která se nachází na první pozici prvního řádku tabulky. Crawler poté kontroluje, zda se zadaná zkratka již nachází v seznamu předmětů – pokud ano, pokračuje na další předmět ze seznamu URL předmětů.

Ze stejného řádku tabulky je pak vybrán i celý název předmětu udaný ve druhém sloupci. Poté je procházen třetí řádek, ve kterém se nachází informace o jméně katedry (zde udáno číslem, později je jméno katedry přiřazeno v crawleru) a počtu kreditů pro tento předmět. Pro hledaný výraz (např.

„ECTS Kredity“) byl v tabulce „manuálně“ dopočten sloupec, ve kterém se výraz nachází, a díky této informaci je poté crawlerem vyselektován.

Následně je pomocí příkazu *select* vybrán první element obsahující slovo *Anotace*. Právě ten obsahuje tuto hledanou část dokumentu. Obsažen je v prostém tagu <*p*>. Dále tak stačí funkcí *.text* extrahovat čistý text a anotace jsou získány.

Poté už stačí získat osnovy přednášek předmětu. Zde opět vzniká problém, protože některé osnovy jsou obsahem tabulky, jiné mají svůj popis uložený jako prostý text. Nejprve tedy proběhne pokus získat element, který obsahuje text „Osnovy přednášek“ následovaný tagem <*table*>. Poté proběhně výběr prvního podřazeného elementu, ze kterého je opět extrahován text, čímž dostáváme i poslední položku, tedy osnovy přednášek.

A.2 Masarykova univerzita Brno

V případě této univerzity byl pro příklad vybrán předmět *Grafický design III* s detaily na adrese `https://is.muni.cz/predmet/fi/podzim2016/PV100`.

Nejprve dochází k získání názvu a zkratky předmětu. Tyto informace jsou uloženy ve společném tagu <*h2*>, který je druhým nadpisem této velikosti na webu. Dochází tak k selekci druhého elementu jmenovaného typu a následnému *parsování* nadpisu. Ten je rozdělen podle mezer, kdy první z částí tvoří zkratku předmětu, z ostatních je sestaven celý název předmětu. Následuje kontrola přitomnosti zkratky v seznamu již získaných předmětů.

Katedra předmětu je zde uvedena v sekci „Garance“, lze však přímo vyselektovat první odkaz v hierarchii dokumentu, který obsahuje právě slovo *Katedra*. Extrakcí textu tohoto odkazu je získáno celé jméno katedry a fakulty odddělené pomlčkou. Poté už stačí tento text opět rozparsovat a vybrat část s názvem katedry.

Kreditové ohodnocení předmětu se nachází pod klíčovým slovem „Rozsah“ (zde je dobře vidět, že každá univerzita si vytváří obsah „po svém“, bez společných konvencí). Toto slovo je tedy opět vyhledáno mezi elementy a poté je vybrán element jemu následující, který obsahuje samotnou informaci o počtu kreditů. Tato informace se zde však opět nenachází sama o sobě a je tak nutné ji ze získaného textu dále parsovat

Jako anotace k předmětu zde byl zvolen text pod pojmem „Cíle předmětu“, přesné slovo „anotace“ se zde nenachází. Je tedy vybrán element obsahující spojení „Cíle předmětu“ a extrahován jeho text, který je následně v crawleru uložen jako anotace.

Osnovy předmětu jsou v tomto případě vyhledávány jako element obsahující slovo „Osnova“. Obsahem elementu je pak nečíslovaný seznam (<*ul*>), který je vybrán. Následně jsou procházeny jeho jednotlivé položky, z nichž je extrahován samotný text osnov a zároveň skládán do výsledného řetězce. Ten je pak crawlerem uložen do objektu předmětu.

A.3 Technická univerzita Liberec

Tato univerzita patří spolu s univerzitou Tomáše Bati, univerzitou Pardubice a Západočeskou univerzitou k těm, které využívají jednotného systému ECTS (jak bylo již zmíněno v popisu crawleru v sekci 8.1.4 na straně 43). Mají tak jednotnou strukturu stránek, stejně jako hierarchii elementů popisující stránku konkrétního předmětu. Vyjímkou je zde ZČU, u které byla shledána menší odlišnost, kvůli které se však musí web procházet trochu odlišným postupem. Více viz sekce A.10 této kapitoly, která se zabývá ZČU.

Ukázkovým předmětem zde bude „Webové aplikace“, dostupný na adrese `http://ects.tul.cz/predmet/NTI/WEAP?lang=cs&rocnik=2&statut=A`.

Základní informace o předmětu se nacházejí v tabulce, která je vyhledána pomocí výrazu *select("table[class=xg_pane]")*. Ten značí tabulku, která obsahuje selektor třídy *xg_pane*. Z této tabulky jsou poté vybírány konkrétní řádky. Hned ten první obsahuje celé jméno předmětu.

Druhý řádek zmíněné tebulky pak obsahuje fakultu a zkratku předmětu, oddělené lomítkem. Získaný text je tedy nutné parsovat na základě znaku „/“. Informace o počtu kreditů přiřazených předmětu se pak nachází na osmém řádku tabulky (ač se na první pohled jeví jako sedmý řádek, je mezi nimi vložen ještě jeden prázný řádek).

Anotace předmětu se v tomto případě nacházejí pod heslem „Obsah předmětu“. Tento obsah je uložen v druhé tabulce hierarchie dokumentu, konkrétně jako její třetí, respektive čtvrtý, řádek – ten obsahuje samotný text anotace (v podobě čistého textu).

Obdobným způsobem jsou pak vybrány i osnovy předmětu (zde uložené pod názvem „Výstupy z učení“). Ty patří pod stejnou tabulku jako výše zmíněné anotace. Text osnov se však nachází až na osmém řádku tabulky.

A.4 Technická univerzita Ostrava

Předmět *Programovací jazyky I*, který byl zvolený jako reprezentant této univerzity, lze dohledat v informačním systému univerzity na adrese `https://www.vsb.cz/cs/studenti/studijni-programy/home`. Přesný odkaz není uveden, protože URL obsahuje mnoho parametrů a je příliš dlouhé.

Název a zkratka předmětu jsou zde obsaženy hned v prvním nadpise webu (tag *<h1>*). Ten je tedy vybrán a dochází k parsování této části crawlerem. Jméno předmětu se vždy nachází před zkratkou, která je uvedena v závorce za předmětem. Na základě závorky je tedy úvodní text rozdělen a je vybrána první část. Ta je dále dělena pomlčkou, v jejíž první části se nachází kód předmětu, v části druhé pak již samotný název.

Zkratka předmětu je získána obtížněji. Bylo nutné sestavit regulární výraz, který je poté aplikován na získaný text a pomocí kterého je vybrána poslední závorka v textu – v té je uvedena zkratka předmětu. Ze závorky je pak extrahována samotná zkratka. Regulární výraz pro tuto extrakci vypadá v crawleru následovně:

```
...
Pattern pattern = Pattern.compile("\\(([^\\(\\)]*)\\)$");
...
```

Informace o katedře a počtu kreditů předmětu jsou pak obsaženy v první tabulce obsahující atribut třídy „*panelGrid detail*“. V jejím prvním řádku, druhém sloupci, se nachází název katedry, ve čtvrtém sloupci pak počet kreditů.

Anotace lze naopak získat jednoduše. Stačí v dokumentu vyhledat první nadpis *<h3>* obsahující přímo slovo „anotace“ a jeho následný * * tag. Ten pak obsahuje čistě text anotace předmětu.

Stejným způsobem je získán i text osnov předmětu – vyhledáním nadpisu a jeho následného tagu, nyní pod pojmem „osnova“.

A.5 Univerzita Karlova

Jako příklad byl zvolen předmět *Datové struktury I*, dostupný na `http://is.cuni.cz/studium/predmety/index.php?do=predmet&kod=NTIN066`.

Název a zkratka předmětu (odděleny pomlčkou) se nachází uvnitř tagu <*div*>, který obsahuje třídu *form_div_title*. Zde je pouze nutno ohlídat případ, kdy předmět samotný obsahuje v názvu pomlčku. Pokud je tedy velikost pole vzniklého po parsování (na základě pomlčky) menší než 2, značí první část název předmětu, druhá část zkratku. Pokud je velikost větší, je nutné ze vzniklých částí sestavit název předmětu a poslední část použít jako zkratku.

Katedru předmětu lze pak získat vyhledáním druhé tabulky v pořadí, obsahující třídu *tab2*. V této tabulce se na druhém řádku nachází název katedry, na řádku šestém naopak počet kreditů pro daný předmět.

Anotace lze získat nelezením tagu <*div*> se třídou *pamela_A_CZE*, který obsahuje samotný text anotace.

Osnovy předmětu zde lze najít pod názvem „Sylabus“, ze kterého lze získat informace podobně jako u anotací. Je vyhledán stejný tag, tentokrát však se třídou *pamela_S_CZE*, ze kterého je získán text osnov předmětu.

A.6 Univerzita Pardubice

Tato univerzita využívá stejného systému jako dříve zmíněná univerzita v Liberci. Popis struktury webu si tedy lze přečíst v sekci A.3 na straně 92.

A.7 Univerzita Tomáše Bati ve Zlíně

Stejně jako předchozí univerzita, využívá i tato systému ECTS. Struktura stránek je tedy shodná s univerzitou v Liberci (sekce A.3 na straně 92).

A.8 Vysoká škola ekonomická v Praze

Pro ukázku struktury webu této univerzity byl vybrán předmět „Informatika“ (konkrétní URL uvedeno není, protože obsahuje mnoho parametrů a odkaz je příliš dlouhý). Předmět lze však dohledat přes katalog na adrese `https://insis.vse.cz/katalog/index.pl`.

Pro získání zkratky předmětu je nutné vyhledat první tabulku v hierarchii dokumentu, ve které je následně vyhledán sloupec tabulky obsahující spojení „Kód předmětu“. Sloupec následující za tímto vyhledaným pak obsahuje v textové podobě zkratku předmětu. Následuje kontrola, zda se zkratka nachází v již „prohledaných“ předmětech. Zde je opodstatnění ještě větší než u ostatních univerzit – tato univerzita vyučuje většinu předmětů v letním i zimním semestru (tedy stejný předmět je vylistován dvakrát), kde jedinou výraznou změnou je jiný vyučující pro daný předmět. Popis a kreditové ohodnocení je ve většině případů nezměněno. Proto stačí daný předmět (na základě jeho zkratky) do databáze zařadit pouze jednou.

Katedra pro tuto univerzitu není na stránce žádného z předmětů uvedena, proto je v crawleru její hodnota nastavena jako „neuvedeno“.

Název předmětu je získán opět z první tabulky webu, kdy je vyhledán sloupec obsahující text „Název česky“, na jehož základě je pak získán název předmětu. Názvy se zde většinou uvádějí v jazyce výuky a anglickém jazyce. Původně bylo zamýšleno tento název získat na základě jazyka výuky, avšak bylo zjištěno, že ne každý předmět tuto položku obsahuje, narozdíl od zmíněného názvu v češtině.

Počet kreditů je opět získán na základě názvu sloupce první tabulky, pod textem „Počet přidělených ECTS kreditů“. Získaný text je nutné ještě následně parsovat, přesný počet kreditů se nachází na prvním místě získaného řetězce, respektive před závorkou v tomto řetězci.

Anotace o předmětu jsou uloženy pod heslem „Zaměření předmětu“, které je vyhledáno v řádku tabulky. Poté již stačí získat prostý text celého řádku. Stejně tak se získají osnovy předmětu, udávané pod názvem „Obsah předmětu“.

A.9 Vysoké učení technické v Brně

K popisu struktury webu této univerzity byl vybrán předmět „Petriho sítě“, s detaily dostupný na `https://www.vutbr.cz/studium/ects-katalog/detail-predmetu?apid=174569`.

Název předmětu lze dohledat pod prvním nadpisem druhé urovně, tedy tagem *<h2>*. Zde se nachází jako prostý text.

Pro vyhledání zkratky předmětu je nutné najít první tabulku na webu předmětu. V jejím prvním řádku se nachází název katedry a zraktra předmětu odděleny pomlčkou, je tedy nutné získaný text rozparsovat. Výsledkem jsou pak tyto dvě hodnoty.

Počet kreditů je uveden v té samé tabulce, dohledán je jednoduše pomocí výrazu „Počet kreditů“. Zde se vyskytuje jedna zvláštnost – některé předměty mají počet kreditů uveden jako „0?“. V takovém případě je tato hodnota crawlerem kontrolována. Pokud se zde tato hodnota nenachází, je z textu získána číselná hodnota počtu kreditů. V opačném případě je tato hodnota nastavena na 0.

Anotace předmětu jsou pak k dohledání v první tabulce se třídou *data*. V té je zároveň vyhledáno spojení „Obsah předmětu (anotace)“. V tagu *<blockquote>* se pak nachází text anotací. Bohužel však po detailnějším prohlédnutí více předmětů bylo zjištěno, že žádný z nich neobsahoval text anotací. U všech bylo uvedeno pouze věta „Není specifikováno.“.

Stejný případ pak platí pro osnovy předmětu. Pod tabulkou se stejnou třídou, jako výše zmíněné anotace, která obsahuje slovní spojení „Osnovy výuky“, se nachází text osnov. Avšak i v tomto případě nejsou osnovy u žádného z předmětů uvedeny. To je problém jak pro danou univerzitu, tak pro zájemce o studium na této univerzitě. Důsledkem je totiž to, že pokud nebude uživatel vyhledávat pouze jména předmětů, velmi pravděpodobně nezíská žádné výsledy z této univerzity, když není uveden žádný text anotací ani osnov.

A.10 Západočeská univerzita

Západočeská univerzita opět využívá stejného systému jako univerzita v Liberci (A.3 na straně 92). Avšak při posledních zkouškách crawleru zde došlo k chybě, některé informace nebyly při crawlingu získány. Není jisté, zda bylo toto dříve přehlédnuto (při stahování dat ze všech univerzit najednou je přehlednutí snadné), či došlo v nedávné době ke změně struktury stránek předmětů. Na základě tohoto faktu byl však crawler náležitě upraven a nyní již získává správné informace i pro ZČU.

Získání zkratky předmětu, jeho názvu, katedry a počtu kreditů je shodné s popisem Liberecké univerzity. Změna byla dohledána pouze ve struktuře anotací a osnov předmětu.

Anotace předmětu se zde nachází v druhé tabulce hierarchie webu. Její čtvrtý řádek obsahuje text uvedený pod nadpisem „Obsah předmětu“ – ten zde byl zvolen jako anotace daného předmětu.

Pro osnovy předmětu byla vybrána sekce pod heslem „Výsledky učení“. Zde již vyvstal první problém – tato sekce je pouze u některých předmětů dělena na další části, konkrétně „Odborné znalosti“ a „Odborné dovednosti“. Pokud se zde tyto dva výrazy nachází, je brána jako osnovy předmětu část „Odborné znalost“. Jednotlivé položky této části jsou v dokumentu sestaveny jako řádky tabulky, proto bylo nejprve nutné zjistit, kolik řádků textu se nachází mezi řádky obsahující slova „znalosti“ a „dovednosti“. Následně jsou pak požadované řádky v cyklu procházeny a jejich text extrahován a poskládán do jednoho výsledného řetězce.

Pokud se výše uvedené řádky „znalosti“ a „dovednosti“ v tabulce nenacházejí, pak je brán text pouze mezi řádky „Výsledky učení“ a „Hodnotící metody“.

Tento přístup k tvorbě stránek je dle mého názoru špatný. Jednotlivé weby s informacemi o předmětech mají různou strukturu, různé počty uvedených údajů, text je do kódu zadán jako řádky jedné tebulky (mnohem vhodnější by bylo do jediného řádku vložit například nečíslovaný seznam, který je přehlednější a snazší pro parsování) atd. To velmi ztěžuje naprogramování crawleru, nikdy není dopředu jasné, jakou strukturu bude mít následující předmět v seznamu.

B Uživatelská příručka

Tato příloha slouží jako návod pro uživatele, jakým způsobem tuto práci používat. Uvedeny budou požadavky pro spuštění této práce, instalace a zprovoznění jejích jednotlivých částí a závěrem pak popis používání práce. Tato příručka je napsána pro prostředí OS Windows 10, práce však byla testována i na systémech založených na OS Unix, kde nebyl problém s její funkčností. Rovněž je v textu předpokládáno, že má uživatel práva administrátora a může tedy spouštět programy a zapisovat do všech potřebných složek v systému.

B.1 Požadavky

Pro spuštění a správnou dunkčnost crawleru je nutné mít nainstalované prostředí *Java SE Runtime Environment 8*. Toto platí i pro úložiště, u kterého je dále nutné počítat s minimálně 51 MB prostoru disku na počítači (respektive alespoň 10 MB pro data stažená crawlerem, zbytek zabírají soubory úložiště). Posledním požadavkem pro správné zprovoznění práce je pak webový server *Apache Tomcat* verze 7. Pro použití crawleru je rovněž nutné mít přístup k Internetu.

B.2 Instalace

Popis instalace, respektive zprovoznění, bude uveden pro jednotlivé části zvlášť. Aby byl výsledek plně fungující, je nutné provést všechny níže zmíněné části.

B.2.1 Crawler

V případě crawleru není nutné nic instalovat. Crawler je publikován jako archivační soubor JAR, který stačí pouze spustit (bude popsáno dále).

B.2.2 Úložiště

V tomto případě se jedná o úložiště *ElasticSearch*, využívané ve verzi 5.0.0. To je nutné stáhnout ze stránek jeho výrobce, konkrétně na adrese `https://www.elastic.co/downloads/past-releases/elasticsearch-5-0-0`. Zde se nachází jako archiv ZIP (lze však stáhnout libovolný z archivů). Stažený archiv je poté nutno rozbalit na disk počítače, např. do složky „C:\ElasticSearch“.

Pro následné nastavení úložiště je pak nutné stáhnout si grafické prostředí *Kibana*. To lze ve verzi 5.0.0. stáhnout z následujícího odkazu: `https://www.elastic.co/downloads/past-releases/kibana-5-0-0`. Pod položkou s názvem „WINDOWS“ se opět nachází archiv ZIP. Ten stačí po stažení opět rozbalit do libovolné složky v počítači, např. do „C:\Kibana“. Více k instalaci a nastavování úložiště není potřeba dělat.

B.2.3 Server

Aplikace serveru je publikována jako archivační soubor WAR. K jeho spuštění je nutné nainstalovat server Apache Tomcat 7. Ze stránek na adrese `http://tomcat.apache.org/download-70.cgi#7.0.77` je opět možno tento server stáhnout, například v podobě archivu ZIP. Ten je následně nutné rozbalit, např. do složky „C:\Tomcat7“. Tím je server „nainstalován“, spuštění aplikace se bude věnovat další část textu.

B.3 Spuštění a obsluha

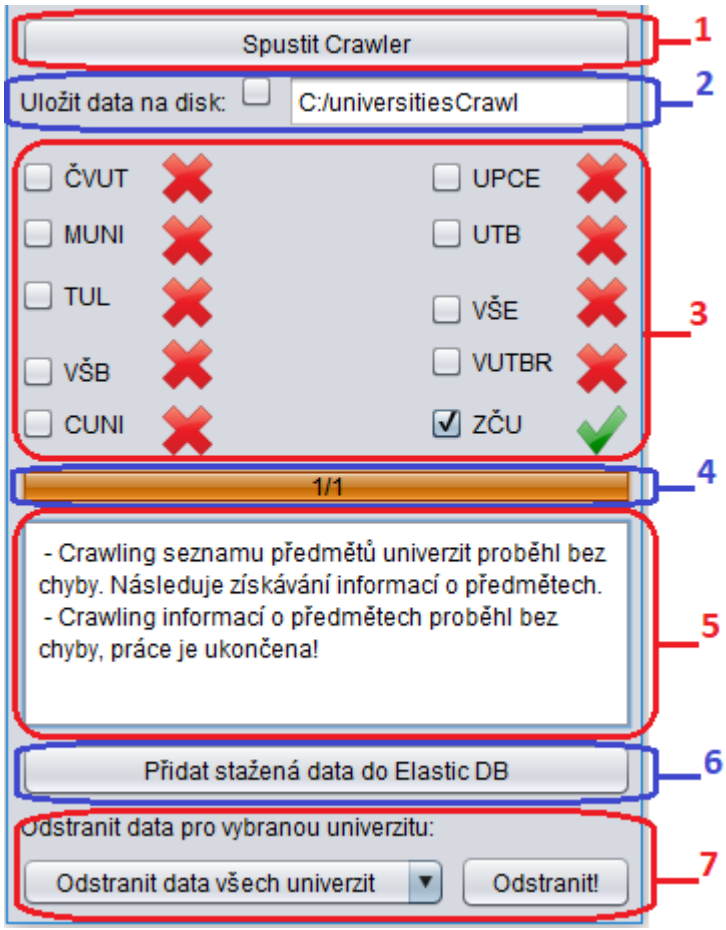
Tato část již konkrétně rozebere, jakým způsobem spustit a ovládat jednotlivé části výsledné práce.

B.3.1 Crawler

Spuštění crawleru je prosté. Ve zmíněném systému Windows stačí poklepak na soubor crawleru, pojmenovaný jako *DPUniversitiesCrawler.jar*. Případně je možné soubor spustit přes příkazový řádek. Ten je nutné přepnout do

pracovního adresáře, kde je archiv crawleru uložen a příkazem „*java -jar DPUniversitiesCrawler.jar*“ crawler spustit. Před jeho spuštěním je ještě možné nastavit konfigurační soubor pojmenovaný „config.json“, který musí být přiložen ve stejné složce jako spouštěný JAR soubor crawleru. V tomto konfiguračním souboru lze nastavit informace pro připojení crawleru v databázi, avšak ve výchozím nastavení (pokud uživatel postupuje podle této příručky) není potřeba nic měnit. Zmíněný soubor vlastně ani není potřebný – pokud se ve stejné složce jako crawler nenachází, jsou implicitní hodnoty uloženy přímo v kódu crawleru.

Po spuštění se zobrazí grafické rozhraní crawleru, viz obrázek B.1, na kterém je zobrazono GUI, v němž již proběhl crawling univerzity ZČU. Tento obrázek má očíslované jednotlivé níže popisované části. Stejný obrázek, bez tohoto popisu, lze nalézt na straně 39 pod číslem 8.6.



Obrázek B.1: Rozhraní crawleru

Ovládání crawleru je pak jednoduché – nejprve je možné zvolit si konkrétní univerzity, pro které bude crawler získávat informace (na obrázku červeně označená část 3). Implicitně jsou zvoleny všechny univerzity, pro zrušení výběru některé z nich stačí odškrtnout tlačítko, které se nachází po levé straně zkratky univerzity. V případě, že si uživatel přeje uložit stažená data na disk počítače, je nutné zaškrtnout tuto volbu příslušným tlačítkem, které se nachází po pravé straně textu „Uložit data na disk:“ (modrá část 2). Vpravo od tohoto tlačítka je navíc textové pole, do kterého lze zadat cestu v souborovém systému, kam budou data uložena – implicitně je zde hodnota „C:\universitiesCrawl“. V té jsou následně data uložena, pro každou univerzitu je vytvořena zvláštní složka, která pak ještě obsahuje složku katedry, ve které se již nachází samotná stažená data. Tuto volbu však není nutné zaškrtávat, slouží pouze pro účely archivace dat, pro tuto práci není nezbytná.

Následuje pak samotné spuštění procesu crawlingu, které se provede stisknutím tlačítka v horní části rozhraní (pojmenovaném „Spustit Crawler“, červeně označená část 1). Crawler započne svou činnost, v tuto chvíli by uživatel měl pouze čekat na výsledky crawlingu. Jeho kroky jsou vypisovány v prostřední části rozhraní určené pro výpisy hlášení (červená část 5). Stahování dat by mělo trvat přibližně 5 minut. Pokud v této fázi dojde k nějaké chybě, je vypsána ve zmíněné prostřední části. Postup crawleru lze vizuálně sledovat na „progress baru“ (modrá část 4) umístěném pod zkratkami univerzit – ten zobrazuje, kolik práce již bylo dokončeno. V případě, že crawling vybraných univerzit proběhl bez problémů, je zároveň vpravo od zkratky univerzity (v části kde probíhá jejich výběr, část číslo 3 z obrázku) změněn červený křížek na zelenou „fajfku“. Pokud jsou všechny vybrané univerzity „označeny zeleně“ a výpis nehlásil žádnou chybu, proběhl crawling úspěšně.

Pokud však některá z univerzit zahlásila chybu, respektive její crawling nedoběhl úspěšně, je vhodné zkontrolovat internetové připojení a zkusit spustit crawler pro danou univerzitu znovu. Občas může dojít k tomu, že webové stránky dané univerzity jsou „přetíženy“ a crawleru se v časovém limitu nepodaří web „kontaktovat“, což vede k chybě. V případě, že ani opětovné spuštění crawleru nepomohlo, bude pravděpodobně nutné upravit kód crawleru, respektive kontaktovat administrátora, který úpravu provede.

Po ukončení stažení je možno přidat získaná data do úložiště (viz modrá část 6 z obrázku), které je však nejprve nutno spustit a řádně nastavit (viz sekce B.3.2 na straně 102). Je-li spuštěno, lze stiskem tlačítka s názvem „Přidat stažená data do Elastic DB“ jednoduše přidat stažená data. Tento

proces opět nějakou chvíli potrvá, záleží na objemu stažených dat.

Pro odstranění dat z databáze pak poslouží tlačítko „Odstranit!“ (červená část s číslem 7). Vlevo od něj se nachází nabídka, ve které je možné vybrat některou z konkrétních univerzit, či ponechat implicitní možnost „Odstranit data všech univerzit“. Po výběru požadované možnosti pak stačí stisknout tlačítko pro odstranění dat. To ještě spustí nabídku, ve které je uživatel dotázán, zda si opravdu přeje data odstranit. Po odsouhlasení pak dochází k mazání dat z úložiště (které je opět nutné mít spuštěné, jak bylo zmíněno výše).

B.3.2 Úložiště

Spuštění úložiště je nutné provést přes příkazovou řádku systému. Nejprve je nutné v příkazové řádce nastavit pracovní adresář, ve kterém je úložiště rozbaleno. Složka *bin* pak obsahuje skripty pro spuštění úložiště. Po přepnutí do tohoto adresáře poté stačí zadat příkaz „elasticsearch.bat“. Celý příkaz spuštění může vypadat například následovně: „C:\ElasticSearch\elasticsearch-5.0.0\bin>elasticsearch.bat“. Jeho spuštění pak probíhá přibližně jednu minutu (záleží na rychlosti počítače), průběh lze sledovat v příkazové řádce, ve které bylo úložiště spuštěno. Po ukončení výpisu všech hlášení (přestanou přibývat nové výpisy) je úložiště připraveno k použití.

Úložiště lze po skončení používání vypnout „terminováním“ jeho procesu v příkazové řádce, přes kterou bylo úložiště spuštěno. Stačí stisknout kombinaci kláves „CTLR + c“ a poté písmeno „y“, které potvrdí, že má dojít k „uzavření“ úložiště.

Potřeba je však ještě nastavit indexování a vyhledávání v úložišti přes prostředí *Kibana*. Toto prostředí lze spustit obdobně jako zmíněný *Elastic-Search*. V příkazové řádce je nutné se přepnout do složky, do které byl rozbalen nástroj *Kibana*. V ní se nachází další složka, *bin*. Po jejím otevření (respektive přepnutí se do jejího adresního prostoru) pak stačí spustit příkaz *kibana.bat*. Celý příkaz tedy může vypadat následovně: „C:\Kibana\kibana-5.0.0\bin>kibana.bat“. Po spuštění, které trvá přibližně jednu minutu, opět v závislosti na rychlosti počítače, je *Kibana* připravena k použití. Stačí otevřít libovolný internetový prohlížeč a zadat v něm adresu `http://localhost:5601/`. Naskočí obrazovka (viz obrázek 5.1 na straně 19, v textu práce), na které v levém menu vybereme možnost „Dev Tools“. Následně se otevře obrazovka se dvěma sloupečky, kde do levého budou zadávány všechny následující

příkazy.

Nejprve je nutné vytvořit index pro univerzity, pojmenovaný „universities“. Napíšeme tedy následující příkaz:

```
PUT /universities {
    ”settings”: {
      ”index.number_of_shards”: 1,
      ”index.number_of_replicas”: 0 } }
```

Zmíněný příkaz odešleme kiknutím na zelenou šipku, která se objeví napravo na první řádce tohoto příkazu, když je označen. Odpovědí od databáze je pak „”acknowledged”: true, ”shards_acknowledged”: true“.

Další příkaz je nutné provést, aby bylo možno nastavit databázi na vyhledávání v češtině. Napíšeme tedy příkaz „POST /universities/_close“ a opět odešleme. Následuje složitější příkaz, který vypadá takto:

```
PUT /universities/_settings {
    ”settings”: {
      ”analysis”: {
        ”filter”: {
          ”czech_stop”: {
            ”type”: ”stop”,
            ”stopwords”: ”_czech_”},
          ”czech_stemmer”: {
            ”type”: ”stemmer”,
            ”language”: ”czech”} },
        ”analyzer”: {
          ”czech”: {
            ”tokenizer”: ”standard”,
            ”filter”: [ ”                lowercase”,
              ”czech_stop”,
              ”czech_stemmer”] } } } } }
```

Jeho odesláním dostáváme již známou odpověď „”acknowledged”: true“. Poté lze index opět otevřít, příkazem „POST /universities/_open“, který odešleme. Tím by mělo být uložiště plně připraveno na nahrávání stažených dat z crawleru. V tuto chvíli lze *Kibanu* zavřít, respektive „terminovat“ její

proces stejně jako v případě *ElasticSearch*, který byl zmíněn výše.

V konfiguračním souboru úložiště není nutné měnit žádnou z implicitních hodnot. Vše potřebné je nastaveno od základu a crawler s webovou aplikací rovněž s tímto základním nastavením počítají.

B.3.3 Server a webová aplikace

Pro spuštění webové aplikace je nutné zapnout Tomcat server. Přepneme se tedy do pracovního adresáře, kde se nachází rozbalený archiv serveru. V něm se nachází složka „webapps“, do které nakopírujeme WAR soubor webové aplikace (její název je „DPElasticServer.war“). V kořenovém adresáři rozbaleného Tomcatu se dále nachází složka „bin“. V nové příkazové řadce se opět přepneme do jejího pracovního adresáře. Poté stačí spustit soubor „startup.bat“. Ten zajistí rozbalení a nasazení dříve zkopírovaného WAR souboru. Celý příkaz může vypadat např. takto: „C:\TomCat7\bin>startup.bat“. Ukončení se pak provádí spuštěním „shutdown.bat“.

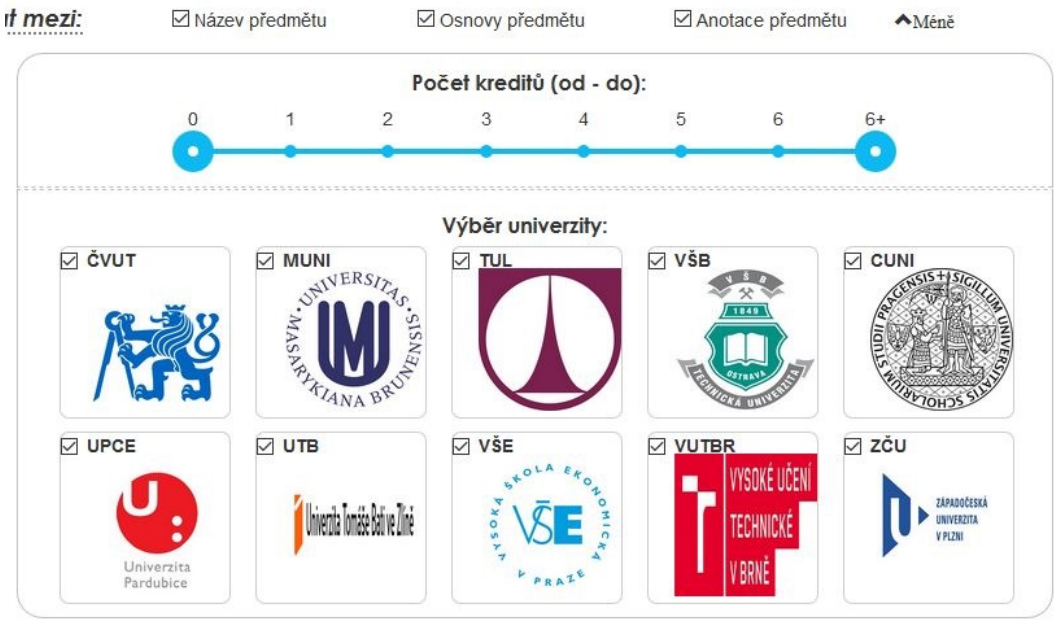
Další nastavení není třeba provádět, po spuštění Tomcatu stačí otevřít libovolný internetový prohlížeč a zadat do něj adresu `http://localhost:8080/DPElasticServer/`. Pokud spuštění serveru proběhlo dle pokynů a bez chyby, měla by být zobrazena webová aplikace, respektive webová část této diplomové práce. Stránka by měla vypada podobně, jako ukazuje její výřez na obrázku B.2.

Obrázek B.2: Výřez z webu po jeho načtení

Použití aplikace je pak prosté. Nejprve si uživatel může zvolit, ve kterých informacích budou vyhledávány jeho dotazy, konkrétně si lze zvolit z možností „název předmětu“, „osnovy předmětu“ a „anotace předmětu“. Výběr probíhá prostým klepnutím na některý z těchto názvů. Implicitně jsou všechny tyto možnosti zvoleny, vždy je však nutné mít zvolenou alespoň jednu z možností.

Po pravé straně se pak nachází šipka, pomocí které lze „rozbalit“ více možností pro konkretizaci hledaných předmětů.

Tyto možnosti si lze prohlédnout na obrázku B.3 níže. Nachází se zde posuvník se dvěma posuvnými prvky, které slouží pro výběr rozsahu kreditového ohodnocení, které budou vyhledávané předměty splňovat. Hodnota „6+“ zde značí 6 a více kreditů. Pod posuvníkem jsou pak uvedené aktuálně dostupné univerzity, mezi kterými lze prohledávat. Klepnutím na logo některé z univerzit dojde k jejímu „odznačení“, což znamená, že z této univerzity nebudou nabízeny žádné předměty. Opět je nutné mít zvolenou alespoň jednu z univerzit, aby bylo možné vyhledávat.



Obrázek B.3: Zobrazené dodatečné možnosti

Po zvolení zmíněných možností je pak možné zadat hledaný výraz. Ten musí obsahovat alespoň tři znaky a zadáván je do textového pole, které se nachází po levé straně tlačítka s textem „Vyhledat“ (viz obrázek B.2 uvedený výše). Zadaný výraz by (v aktuální verzi této práce) měl cílit na oblast počítačů a problematiky s ní spojené. Příkladem budiž dotaz „programování v jazyce C“. Jeho odeslání pak probíhá zmíněným tlačítkem „Vyhledat“.

Jak je zmíněno v odstavci výše, je možné zadávat i víceslovné dotazy. Vyhledávání v databázi pak probíhá takovým způsobem, že je dotaz rozdělen na jednotlivá slova, která jsou poté ve vybraných polích (anotace atd.) vy-

hledávána samostatně. Více o metrice, jakým způsobem jsou vyhledané výrazy vyhodnoceny, si lze přečíst v textu práce (závěr sekce 8.3.2 na straně 59). Aktuálně není možné žádný z hledaných výrazů uživatelem zvýhodňovat, například jeho označením uvozovkami, jako umožňují některé internetové vyhledávače. Jednotlivá slova složeného výrazu jsou pak prohledávána jejich složením pomocí klauzule *OR*, což znamená, že pokud se v hledaném poli nachází alespoň jedna z částí výrazu, je toto pole akceptováno.

Pokud se hledaný výraz v úložišti nachází, jsou v levé části webu zobrazeny výsledky, respektive názvy předmětů odpovídajících hledanému dotazu. U nich je ještě uvedeno „Skóre“, které udává shodu hledaného výrazu se záznamy v úložišti – slouží spíše pro rychlé porovnání, který z předmětů lépe odpovídá zadání (pro přesnější popis výpočtu skóre viz závěr sekce 8.3.2 na straně 59). Jak tyto výsledky vypadají ukazuje obrázek B.4.

Obrázek B.4: Zobrazení výsledků vyhledávání

Jednotlivé názvy vyhledaných předmětů si lze následně rozkliknout (kliknutím do oblasti s názvem předmětu). Tím dojde k zobrazení dodatečných informací pro tento předmět. Na základě těchto informací se pak uživatel rozhodne, zda se mu obsah předmětu zamlouvá. Pokud ano, stačí vpravo od jména daného předmětu zmáčknout tlačítko se znakem „+“. To zařadí předmět mezi oblíbené, což zároveň změní znak tlačítka na „-“, což umožní

daný předmět opět odebrat.

Pokud by uživateli nestačil zobrazený počet předmětů, je možné kliknout na tlačítko s textem „Načíst další (+10)“, které pro poslední hledaný výraz načte dodatečné výsledky (pokud takové existují). Toto tlačítko se nachází pod seznamem vyhledaných předmětů a jeho stisk jde kvůli přehlednosti opakovat pouze pětkrát, zobrazeno tak může být maximálně 60 předmětů pro jeden výraz.

Po zařazení prvního předmětu mezi oblíbené dojde v levé horní části webu k zobrazení tlačítka „Zobrazit oblíbené“ (nachází se pod „počtem oblíbených předmětů“). Jeho stiskem dojde k přepnutí na druhou část webu. V té se vlevo nachází seznam oblíbených předmětů, který je udělán stejným způsobem jako předměty vyhledané v databázi. Jednotlivé předměty lze pro zobrazení detailů opět rozklikávat. Po pravé straně jejich jména se pak nachází známé tlačítko, tentokrát pouze se znakem „-“, které slouží pro odebrání daného předmětu. Po jeho stisku zmizí předmět z výběru, pro opětovné přidání je nutné ho opět v databázi vyhledat a přidat mezi oblíbené.

Na pravé straně této druhé části webu se pak nachází seznam univerzit a k nim přísušný počet předmětů, které byly u dané univerzity označeny jako oblíbené. Pod tímto seznamem je pak logo doporučené univerzity, vybrané na základě té školy, pro kterou bylo zvoleno nejvíce oblíbených předmětů (pokud je zde shoda a není jasný „vítěz“, je vypsán text „Na základě aktuálního výběru nelze rozhodnout“).

Přepnutí zpět na část s vyhledáváním předmětů lze tlačítkem na stejné pozici (levá horní část webu), kde bylo tlačítko pro zobrazení oblíbených předmětů. V tomto případě se tlačítko jmenuje „Zpět na vyhledávání“.

Po ukončení práce s webovými stránkami stačí jednoduše zavřít okno prohlížeče.