VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKY AKTUALIZOVANÝ WEBOVÝ PORTÁL O EVROPSKÝCH VÝZKUMNÝCH PROJEKTECH

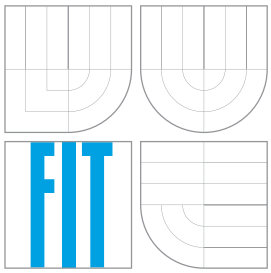
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUCIE DVOŘÁKOVÁ

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKY AKTUALIZOVANÝ WEBOVÝ PORTÁL O EVROPSKÝCH VÝZKUMNÝCH PROJEKTECH

AUTOMATICALLY UPDATED WEB POTRAL ON EUROPEAN RESEARCH PROJECTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUCIE DVOŘÁKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. PAVEL SMRŽ, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Cílem práce je vytvořit webový portál umožňující uživateli vyhledávat evropské výzkumné projekty. V porovnání se stávajícími řešeními se pak specializuje na plnotextové vyhledávání v technických zprávách vydaných v rámci jednotlivých projektů. Součástí práce je také program provádějící automatickou aktualizaci dat portálu.

Abstract

The Bachelor's Thesis aims at creating a web portal allowing users to search within European research projects. It is optimized for full-text searches including project's deliverables. The thesis also includes a tool responsible for automatic update of web portal data.

Klíčová slova

web, portal, Python, Elastisearch, Flask

Keywords

web, portal, Python, Elastisearch, Flask

Citace

Lucie Dvořáková: Automaticky aktualizovaný webový portál
o evropských výzkumných projektech, bakalářská práce, Brno, FIT VUT v Brně, 2015

Automaticky aktualizovaný webový portál o evropských výzkumných projektech

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph. D. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Lucie Dvořáková
17. května 2015

Poděkování

Děkuji vedoucímu této bakalářské práce panu doc. RNDr. Pavlu Smržovi, Ph.D. za aktivní odbornou pomoc a podněty při řešení této práce.

Obsah

Kapitola 1

Úvod

Cílem mé bakalářské práce bylo navrhnout a implementovat automaticky aktualizovaný webový portál o evropských výzkumných projektech. Práce nespočívá pouze ve vytvoření uživatelského rozhraní v podobě webové aplikace, ale také v automatickém získávání dat, návrhu způsobu jejich uložení a optimalizace procesu vyhledávání.

Webové portály zpravidla umožňují uživateli vyhledávat v datech na základně různých kritérií. V případě informací o projektu těmito kritérii mohou být např. jeho název akronym, řešitel, apod. Uživatel si tak snadno může vytvářet různé náhledy, a to jak pomocí jednoduchého intuitivního ovládání, tak pomocí pokročilého dotazování. Portály také zpravidla bývají schopné nalézt souvislosti mezi uloženými informacemi a mohou uživateli poskytovat další na základě jejich podobnosti.

V porovnání se stávajícími řešeními se nově navrhovaný portál také specializuje na efektivní vyhledávání v technických dokumentech vydaných v rámci jednotlivých projektů. Zprávy, které nejsou přidány pracovníkem Evropské komise do specializovaných databází, se však často stávají po ukončení projektu nepřístupnými. Nově navrhovaný portál se snaží tyto zprávy archivovat přímo z webových stránek jednotlivých projektů, a umožňuje v nich tedy vyhledávat i nadále.

V kapitole 2 s názvem Stávající webové portály uvádím příklady běžně využívaných postupů a technologií u existujících webových portálů podobného rázu. Úvod do problematiky automatického získávání dat, plnotextového vyhledávání a návrhu uživatelského rozhraní. Následující kapitola Návrh a implementace se věnuje již samotné tvorbě jednotlivých částí portálu. Nejdříve se kapitola zabývá specifikací požadavků, které slouží k vytvoření uceleného náhledu na vše, co je potřebné vytvořit pro splnění požadavků na webový portál. Specifikace využívám také jako plán, v jakém pořadí bude implementace probíhat a jakým způsobem budou jednotlivé části navazovat. Dále jsou pak teoreticky popsány standardní postupy. Detailněji se věnuji popisu technologií, které zpravidla nejsou běžně využívány. Stejně tak jsou popsány i nestandardní způsoby implementace řešící problémy, které při práci vznikaly. Předposlední kapitola Testování portálu a extraktoru dat popisuje testování správnosti implementace jednotlivých částí webového portálu a jeho extraktoru dat. V závěrečné kapitole je shrnuto zhodnocení odvedené práce a analýza výsledného projektu.

Kapitola 2

Principy využívané u stávajících webových portálů

Webových portálů umožňujících jejich uživatelům vyhledávat ve veřejně přístupných textech dokumentů je dostupné velké množství. Z pohledu zaměření této bakalářské práce jsou relevantní portály a digitální knihovny pracující s publikacemi jako např. Google Scholar, Web of Science, ACM DL, Springer, IEEE Xplore nebo DBLP[1]. Další relevantní skupinou jsou např. informační služby agregující informace o projektech financovaných určitou institucí. Jako příklad takových služeb lze uvést evropský informační portál Cordis (projekty financované z grantů EU), portál projektů NSF (americká grantová agentura), či DFG (německá grantová agentura)[2].

Většina těchto portálů má možnost autorizovaným uživatelům přidávat nové informace, které pak tvoří obsah webu. Jiné portály, např. zmíněný Google Scholar nebo DBLP, tuto vlastnost nemají. Informace tedy automaticky získávají z jiných již potvrzených zdrojů. K tomuto účelu mají nástroj, kterým tato data vyhledá a extrahuje. I když data a funkcionality můžou být rozdílné, základní principy a postupy zůstávají obdobné. Každý z těchto portálů musí mít strukturovanou paměť pro uložení dat, způsob, jak tato data automaticky získává, a systém, pomocí kterého k těmto datům uživatel může přistupovat. Touto problematikou se více zabývají jednotlivé části této kapitoly.

2.1 Extrakce dat

Aby došlo k získání potřebných informací, musí existovat program automaticky hledající data z určitých webových stránek pomocí postupného procházení webových odkazů. Vyhledávací webové portály ho využívají zejména za účelem získávání nových a aktualizaci pozměněných dat. Nalezená data jsou pak ukládána tak, aby vyhledávání v nich bylo znatelně rychlejší než opakované prohledávání jednotlivých webových stránek.

Samotné získávání dat se označuje jako *web scraping* [20, 26]. Tento proces má za účel převést nestrukturovaná data nalézající se na internetu a transformovat je na strukturovaná data, která lze ukládat a analyzovat. Za primitivní formu web scrapingu lze považovat manuální kopírovaní dat a jejich uložení. V případě, že je webová stránka chráněná proti

[1] Google Scholar `http://scholar.google.com`, Web of Science `http://webofscience.com`, ACM DL `http://dl.acm.org`, Springer `http://www.springer.com`, IEEE Xplore `http://www.ieee.org/ieeexplore`, DBLP `http://dblp.uni-trier.de`

[2] Cordis `http://cordis.europa.eu`, NSF `http://www.nsf.gov`, DFG `http://www.dfg.de`

automatickému prohledávání, může to být jediný způsob, jak data získat. Příkladem dalšího způsobu, již automatického, je pomocí regulárních výrazů [4] (angl. regular expression matching). Regulární výrazy se využívají k vyhledávání textu, u kterého není přesně známý tvar nebo může mít více podob. I když je použití regulárních výrazů ve své podstatě jednoduché, lze s nimi dosáhnout při vhodné aplikaci velmi dobrých výsledků.

Např. Google u svých dat získaných data scrapingem nepotřebuje vědět přesný význam, více ho zajímá podstatnost a relevance jednotlivých částí. Ty jsou zdůrazněny pomocí předem určených konstrukcí a HTML značek. Například text nacházející se mezi HTML značky `<h1></h1>` bude brán jako hlavní nadpis stránky. Když se pak podíváme na specializované vyhledávače, ty naopak často potřebují zjistit význam jednotlivých získaných položek, aby mohly nabídnout vyhledávání optimalizované právě pro téma, kterému se věnují. Při data scrapingu pak mohou využívat rozpoznávání sémantických anotací. K tomu se využívají *metadata*, data popisující části dat nebo značky, určující význam textu. Toto značení se může nacházet přímo ve zdrojovém kódu webových stránek nebo v separátních souborech specificky určených k usnadnění získání potřebných dat.

Do kategorie specializovaných vyhledávačů patří již zmíněný webový portál DBLP obsahující jednu z největších bibliografií v oboru počítačových věd. DBLP nenabízí možnost přímého přidávání publikací z důvodu možného zneužití, kvůli kterému by došlo ke ztrátě důvěryhodnosti portálu. Získávání dat tedy probíhá automaticky ze stránek významných vydavatelů, případně ze stránek jednotlivých konferencí. Extraktor dat portálu DBLP se pokouší nalézt speciální XML soubor `dblpsubmission.dtd/dblpsubmission.xsd`, obsahující specifickou XML strukturu, kde je pomocí pevně daných značek k jednotlivým datům přiřazen jejich význam. Níže je uveden příklad obsahu popisovaného souboru XML:

```
<dblpsubmission>
    <proceedings>
        <editor>Tal Rabin</editor>
        <title>Advances in Cryptology - CRYPTO 2010.
          30th Annual Cryptology Conference.</title>
        <publisher>Springer</publisher>
        <year>2010</year>
        .
        .
        .
    </proceedings>
</dblpsubmission>
```

2.2 Plnotextové vyhledávání v databázích

Databáze je uspořádaná množina dat uložená v paměti. Typicky je uspořádána tak, aby byl optimalizovaný přístup k nejvíce potřebným datům a získávání těchto informací bylo co nejrychlejší. Pro práci s databází se zpravidla využívají softwarové aplikace tzv. systémy řízení báze dat (SŘBD), které zprostředkovávají možnost efektivně manipulovat a přistupovat k datům.

Při hledání v dokumentech je zpravidla použito *plnotextové vyhledávání*. Při tomto typu vyhledávání dojde k prozkoumaní každého slova v uloženém dokumentu ve snaze najít slova splňující kritéria vyhledávání. Tato analýza velkého množství textu může být velmi časově náročná, takže jsou vytvářeny SŘBD specializující se na tento druh vyhledávání.

2.2.1 Relační databáze

Většina dnešních SŘBD je založena na relačních modelech. V tomto modelu jsou data ukládána do jedné či více tabulek po řádcích s vlastním unikátním klíčem. Díky možnostem odkazovat na další záznamy můžou vznikat obrovské navzájem propojené struktury tabulek s možností efektivního dotazování napříč databází. Relační databáze jsou často neefektivní při plnotextovém vyhledávání a velmi náročná je škálovatelnost distribuovaných systémů.

Při snaze plnotextově vyhledávat v relačních databázích se vytvářejí komplikované dotazy, které často vrací nepřehledné odpovědi. Většinou nemají jazykovou podporu a nevidí tedy shodu při vyhledávání totožného slova pouze v jiném tvaru. Není zde žádný systém řazení vyhledaných výsledků a vyhledávání je velmi pomalé, protože při každém dotazu dojde ke kompletnímu prohledání databáze.

2.2.2 NoSQL databáze

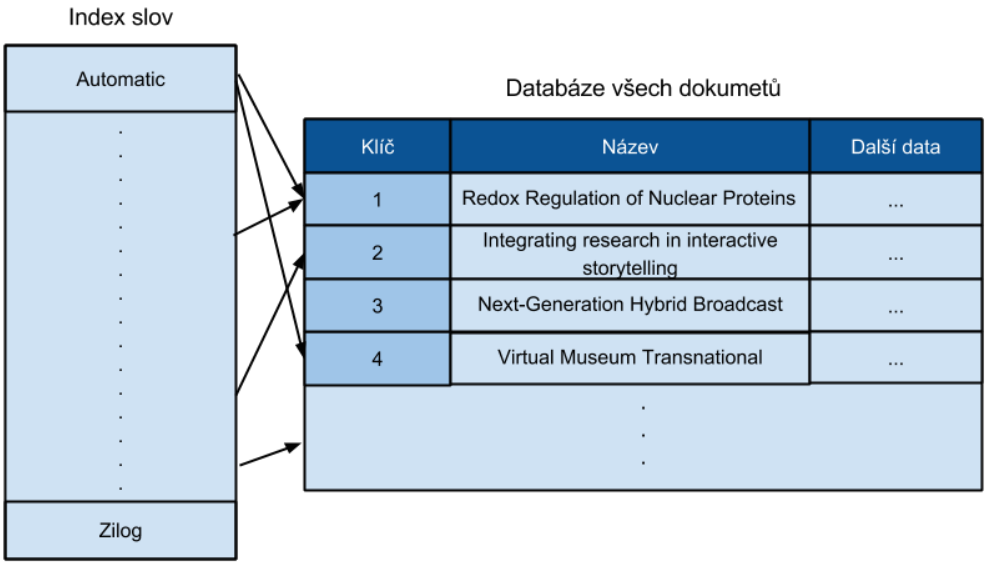
NoSQL („Not only SQL“) SŘBD [1] jsou stále častěji využívány při práci s velkým množstvím dat (angl. big data). Jsou horizontálně škálovatelné, nemají tedy problém s rychlým nárůstem dat ani v rámci distribuovaných systémů. V případě potřeby přidání dalšího serveru dojde k optimálnímu rozložení dat mezi uzly. Dále tento typ SŘBD vyniká při tvorbě webových aplikací běžících v reálném čase, tj. aplikací umožňujících použití vyhledávání a filtrace nad daty s tím, že jsou výsledky zobrazeny prakticky okamžitě.

Z důvodu jejich zaměření bývá struktura NoSQL SŘBD v porovnání např. s dříve prezentovanými relačními SŘBD navržena jinak. Tato struktura silně závisí na typu ukládaných dat [16]. Příklad nejjednoduššího typu NoSQL databáze má strukturu *asociativního pole*. Asociativní pole ukládá data ve formátu klíč-hodnota, kde lze hodnotu dohledat na základě uvedeného unikátního klíče. Existují ale i struktury navržené pro data, která jsou nejlépe reprezentovatelná grafem, jako třeba mapy nebo síťové topologie.

Pro tuto bakalářskou práci je nejrelevantnější struktura databáze specializovaná pro ukládání dokumentů. Každý dokument má svoji unikátní identifikaci a jedno nebo více polí s přiřazenými daty. Ovšem na rozdíl od relační databáze nemusí mít všechny dokumenty svá pole stejná. Není-li určeno jinak, jsou hodnoty jednotlivých polí *tokenizovány*, tj. rozděleny na lexikální jednotky, což umožňuje SŘBD v kombinaci s *indexováním* [17] efektivní plnotextové vyhledávání. Při indexování dochází k vytváření seznamu s jednotlivými slovy či lexikálními jednotkami označovanými jako *index*. Při vložení nového dokumentu do databáze dojde k jeho tokenizaci a uložení jednotlivých výrazů do indexu slov. Společně se slovem se uloží jeho relativní pozice v dokumentu (viz obrázek 2.1). Způsob vyhledávání a filtrování stejně tak jako pokročilé funkce NoSQL SŘBD (např. fasetová klasifikace nebo textové náhledy) jsou v tomto dokumentu dále uvedeny v kontextu SŘBD Elasticsearch [14] použitém pro nově navrhovaný portál.

Kromě pozitivních vlastností jsou použití s použitím specializovaných struktur spojeny i některé problémy jako např. neefektivnost sjednocovacích dotazů [10]. Uvedený problém se nejčastěji řeší podáním několika dotazů za sebou. Jelikož jsou NoSQL dotazy časově méně náročné, je to při omezeném počtu dotazů ideální řešení. Jestliže by však těchto za sebou navazujících dotazů mělo být více, je lepší strukturovat databázi tak, že dojde k denormalizaci spočívající v částečné duplikaci dat. Namísto pole s cizím klíčem, odkazujícím se na jiná data, jsou tato data do pole ukládána přímo.

Elasticsearch je volně dostupný SŘBD postavený na knihovně Apache Lucene [2]. Uve-



Obrázek 2.1: Princip vyhledávání za pomocí slov uložených v indexu.

dená knihovna je využívána mnohými portály velkých institucí[3]. Práce s Elasticsearch přináší již zmíněné NoSQL výhody a další vlastnosti, které jsou praktické při vytváření webové aplikace s plnotextovým vyhledáváním. Komunikace může být zasílána přes protokol HTTP s dotazy zapsanými ve formátu JSON [7]. Elasticseach tak může být využit v kombinaci s jakýmkoliv programovacím jazykem. Mezi vestavěné způsoby dotazování patří například vyhledávání, filtrování, fasetová klasifikace a další.

Vyhledávání a skóre

Vyhledávání v Elasticsearch [11] probíhá pomocí jednoduchého dotazu, který se označuje jako *query*. Níže je uveden příklad dotázání s využitím formátu JSON, který bude pro svou jednoduchost užíván v textu i dále.

```
{
    "query" : {
        "term" : { "objective" : "kinect" }
    }
}
```

Výše uvedený dotaz slouží k nalezení všech záznamů obsahující výraz „kinect“ v poli „objective“. Při vyhledávání záznamů pomocí fráze či klíčového slova přiřadí Elasticsearch ke každému výsledku proměnou s názvem *skóre* [11, 21]. Skóre numericky vyjadřuje relevanci dokumentu k hledanému výrazu a struktura výsledků je seřazena právě podle něj. Dále je uveden příklad možné odpovědi na výše položený dotaz.

```
"hits" : {
    "total" : 170,
```

[3]Např. Mozzila (`www.mozilla.org`), CERN (`home.web.cern.ch`), Netflix (`www.netflix.com`)

```
    "max_score" : 1.0,
    "hits" : [ {
        "_index" : "xdvora1f_projects",
        "_type" : "data",
        "_id" : "6798878",
        "_score" : 0.4,
        "_source":{
            "title": "Integrating research in interactive
                storytelling",
            "programme": "FP7",
            "objective": "The first study focused on full body
                interaction using Microsoft Kinect and ..."
    .
    .
    .
}
```

Jak je možné vidět, Elasticsearch provede analýzu obsahu databáze a vyhodnotí následující informace: kolik shod bylo nalezeno (`total`), jaké maximální relevance k vyhledávání bylo dosaženo (`max_score`) a seznam jednotlivých nalezených záznamů (`hits`). Každý záznam má pak u sebe několik dalších položek jako datový typ (`_data`), unikátní identifikátor (`_id`), již zmíněná relevance daného záznamu k vyhledávání (`_score`) a následovně asociativní pole veškerých informací rozdělených do polí (`_source`).

Při vyhodnocování skóre využívá Elasticsearch několik metod, mezi které patří například *tf-idf* [21]. Výpočet hodnoty *tf-idf*, tj. četnosti termínu-převrácené dokumentové četnosti (angl. term frequency-inverse document frequency), se skládá ze dvou částí. Četnost termínu *tf* je možné odvodit jako poměr frekvence výskytu $f(t, d)$ hledaného termínu t v dokumentu d vůči frekvenci výskytu $f(w, d)$ termínu w s nejvyšší četností.

$$tf(t, d) = 0,5 + \frac{0,5 \times f(t, d)}{max\{f(w, d) : w \in d\}}$$

Druhá část, tzv. převrácená dokumentová četnost *idf*, zohledňuje podstatnost hledaného termínu. Čím častěji se termín t v dané množině dokumentů D vyskytuje, tím menší váha se mu přikládá. (Malá váha bývá z pravidla přiřazena např. anglickým členům „a“ a „the“.)

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Celkově lze potom hodnotu *tf-idf* vypočítat jako součin výše popsaných hodnot.

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t, D)$$

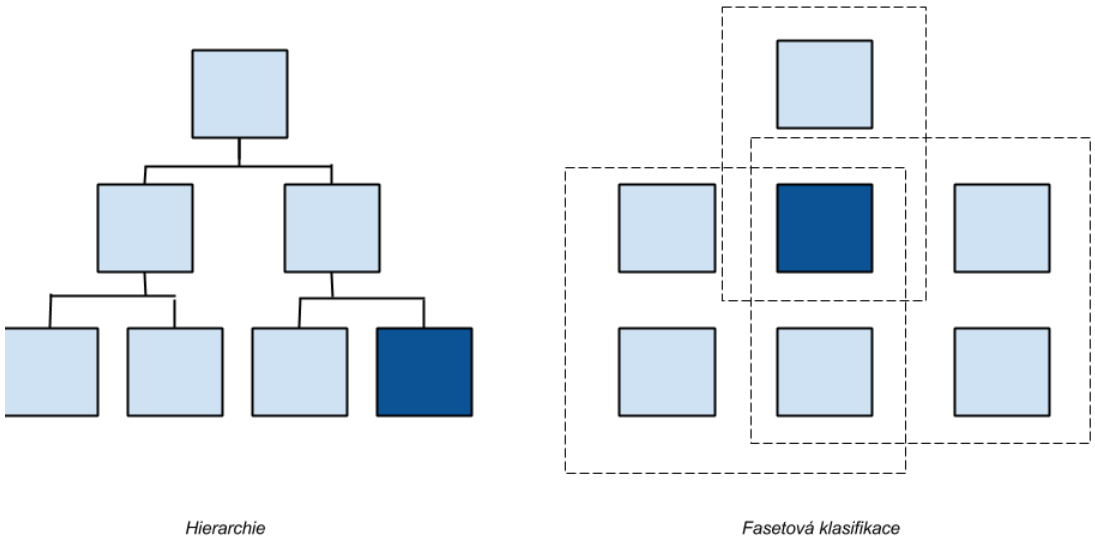
Filtry

Operace výpočtu čísla relevance je časově náročnější, a je-li třeba pouze výběr zúžit bez potřeby řazení, může se využít filtrovací dotaz [8]. Prakticky lze filtrovací dotaz využít k získání všech záznamů se stejnou hodnotou v určeném poli, kde relevance je pak u všech záznamů ve výsledku stejná.

```
{
    "filter" : {
        "term" : { "country" : "Czech Republic"}
    }
}
```

Fasetová klasifikace

Další způsob dotazování v Elasticsearch databázi se označuje jako *fasetování*, pomocí kterého se vytváří tzv. *fasetová klasifikace* [12, 22]. Fasetová klasifikace nabízí možnost zařazení dokumentu do více tříd na základě hodnot jednotlivých polí.



Obrázek 2.2: Schéma pro porovnání hierarchie a fasetové klasifikace.

Při vyhledávání v běžné hierarchii jsou prvky rozděleny a seřazeny podle jedné specifikace. Při fasetové klasifikace lze prvek přiřadit k několika specifikacím. Rozdělení a seřazení bude tedy různé v závislosti na vybrané specifikaci, viz schéma 2.2.

Praktické využití fasetové klasifikace je právě ve webových aplikacích, kde mohou sloužit jako menu pro přesnější specifikaci při vyhledání. Pro příklad, pokud by chtěl uživatel nalézt vědecký článek, o kterém ví, že byl vydán roku 2012 v Německu, může vyhledat průnik množiny článků vydané v roce 2012 a těch, co byly vydány v Německu. Výběr výsledků je tím značně omezen a nalezení hledaného článku je tak jednodušší.

Při použití funkce fasetování ve spojení s Elasticsearch databází se musí nejdříve určit atributy dokumentu, na základě kterých se bude moci specifikovat. Funkce vrátí seznam všech hodnot, které dané atributy obsahují, společně počtem výskytů konkrétní hodnoty. Když je vybrána jedna či více hodnot ze seznamu, dojde pomocí nich k filtrování databáze a výpis už je zhotoven pouze z dokumentů, které požadované specifikace splňují.

Textové náhledy

Textový náhled je ukázka části textu (angl. snippet) sloužící k objasnění uživateli, proč

byl daný dokument při vyhledávání vybrán [9]. Textový náhled obsahuje hledaný pojem a text, který se nachází před a za ním. Tímto napomáhá uživateli zjistit, v jakém kontextu bylo slovo použito. Další funkcionalitou spojenou s textovými náhledy je možnost zvýraznění (angl. *highlighting*). Při jeho použití je nutno definovat, z kterého pole se má textový náhled vytvořit, je-li hledaný výraz nalezen. Mezi nepovinné parametry patří například délka výsledného řetězce a jakým způsobem má hledaný pojem být zvýrazněn. Funkce vrací seznam nalezených řetězců.

Vyhledání podobných dokumentů

Nalezení souvislostí napomáhá nalezení vzájemně podobných dokumentů. Pomocí této metody lze uživateli vytvořit nabídku projektů, u kterých je zvýšená pravděpodobnost, že jej mohou zaujmout. V Elasticsearch je tato metoda známá jako funkce *more-like-this* [13].

Funkce more-like-this vytváří query dotazující se, zda se v dokumentu nachází slova relevantní pro daný dokument. Relevance se určuje na základě již zmíněné hodnoty *tf-idf*, jejíž výpočet lze dále ovlivňovat podmínkami jako např. omezení minimálního počtu výskytů relevantního slova, jak minimálně/maximálně smí být dané slovo dlouhé nebo jaké je minimální procento nalezených relevantních slov v dokumentu, aby byl stále považován za podobný.

Funkce také umožňuje přesně specifikovat pole, dle kterého se má podobnost hledat. Není-li zadáno, hledá se shoda ve všech možných polích. Funkce vrací seznam všech dokumentů, u kterých byla shoda určena jako významná. Součástí výsledku je podobně jako v případě query i skóre značící míru relevance.

2.3 Metodologie tvoření grafických uživatelských rozhraní

Uživatelské rozhraní je vytvářeno za účelem, aby měl běžný uživatel možnost získaná data přehledně zobrazovat. Podstatné je zajistit jednoduchý design a intuitivní ovládání [18].

Na základě zaměření portálu lze odvodit určité ovládací prvky, které si kladou za cíl jeho snadné používání. Běžný uživatel nechce trávit čas hledáním ovládacích prvků nebo snahou pochopit, jak tyto prvky fungují. Jedním z pravidel při vytváření intuitivního rozhraní je tak snaha neměnit standardní rozsazení prvků. Někteří uživatelé chtějí hned po příchodu najít vyhledávací pole bez toho, aby museli projít celý web nebo automaticky klikají na logo webového portálu pro návrat na úvodní stránku. Pro uživatele je jednodušší využívání rozhraní, které již zná, a tedy použití rozmístění prvků podobné jako např. u vyhledávače Google pomůže mnoho uživatelům k jejich rychlému zorientování.

Dále je podstatné uživatele alespoň z počátku nepřehlcovat a nabídnout mu pouze ty možnosti, které bude skutečně potřebovat. V angličtině se tento přístup označuje jako „The principle of good enough“ („Princip dostatečně dobrého“). Zkušenému uživateli lze pak nabídnout další způsoby ovládání urychlující jeho aktivity, jako například vyhledávací pole, které dovoluje přímo specifikovat pokročilé filtry.

Kapitola 3

Návrh a implementace

Po teoretickém úvodu do problematiky týkající se jednotlivých částí bakalářské práce se může přejít k samotnému návrhu a implementaci. Nejdříve se vytvoří ucelená specifikace prvků a funkcionalit, které musí práce obsahovat, aby byl webový portál plně funkční. Dále následuje detailní popis implementace jednotlivých částí práce.

3.1 Specifikace požadavků

Cílem bakalářské práce je vytvořit automaticky aktualizovaný webový portál, avšak před jeho implementací je potřeba vytvoření extraktoru dat a způsob ukládání dat optimalizované pro plnotextové vyhledávání. Je nutné si uvědomit, co vše má výsledný webový portál splňovat a podle toho jednotlivé části implementace směrovat. Dále je popsána implementace extraktoru dat, využití databáze Elasticsearch a ukázání možností při práci s uživatelským rozhraním. Na závěr kapitoly je schématem znázorněna a popsána komunikace mezi jednotlivými částmi projektu.

3.1.1 Požadavky na extraktor dat

Extraktor dat má za cíl získat dostatečné množství dat, aby bylo možné zjistit rychlost vyhledávání podobnou reálnému provozu. Extrakce probíhá z vědeckého webového portálu Cordis, který obsahuje informace o projektech a v mnohých případech dokumenty, které se k danému projektu vztahují. Skript musí projekty vyhledat, extrahovat všechna potřebná data a zaindexovat je do databáze. V případě výskytu některého dokumentu musí být schopen jej stáhnout, otevřít, normalizovat obsah a v textové podobě jej taktéž zaindexovat do databáze. Další důležitá vlastnost extraktoru dat je prohledávání oficiálních webových stran projektů, tedy těch, které se nenachází na portálu Cordis, a nalézt technické zprávy.

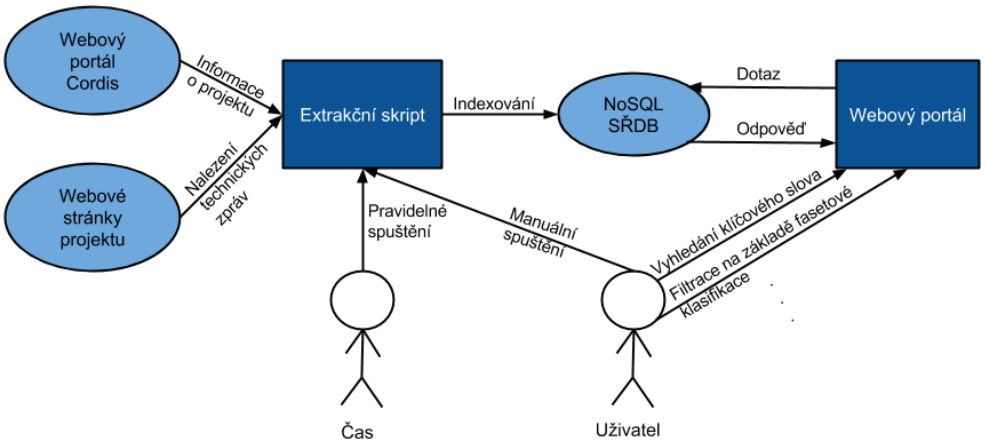
3.1.2 Požadavky na databázi a způsob uložení dat

Po extrakci takového množství dat se jeden z hlavních požadavků na využívanou databázi Elasticsearch stává rychlost. Uživatel klade dotazy přímo přistupující do databáze a ta musí téměř okamžitě uživateli zprostředkovat požadované informace. Základem je vymyslet efektivní způsob propojení projektů s jeho dokumenty s co možná nejmenší redundancí. Další požadavek je vytvoření menu, pomocí kterého uživatel může specifikovat svůj dotaz. K tomu budou využity již zmíněné fasety. Touto vlastností je nutno se zabývat ještě před samotným spuštěním extraktoru, protože fasety nelze vytvářet z dat, které Elasticsearch

ve výchozím nastavení tokenizuje. Více o této problematice pak v kapitole 3.3.1 o struktuře databáze.

3.1.3 Požadavky na návaznost jednotlivých částí

Znázornění interakce jednotlivých částí projektu je patrna na obrázku 3.1. Extrakční skript je spuštěn v určitých časových intervalech pro pravidelnou aktualizaci, ale lze jej spustit i manuálně uživatelem. Skript přijímá data z webového portálu Cordis nebo z webových stránek jednotlivých projektů a přidává je do NoSQL SŘBD (konkrétně Elasticsearch). Odtud jsou požadované informace dále předávány webovému portálu na základě dotazů a požadavků od uživatele.



Obrázek 3.1: Schéma návaznosti jednotlivých částí.

3.1.4 Požadavky na prezentační část

Data je poté nutné prezentovat uživateli tak, aby se k požadovaným projektům dostal co nejrychleji a za pomocí co možná nejmenšího úsilí. To má za úkol uživatelské rozhraní, v našem případě webové uživatelské rozhraní. Musí být strukturované tak, aby byl uživatel schopen se v něm rychle zorientovat a zároveň, aby mělo čistý a nenápadný vzhled.

Je několik ovládacích prvků, které musí uživatelské rozhraní zahrnovat. Jedním z nich je možnost zadání hledaného výrazu, standardně řešené pomocí umožnění jeho zápisu do textového pole. Dalším prvkem je zpravidla menu, které umožňuje uživateli filtrovat data na základě určitých vlastností projektu. Uživateli bude také poskytnuta možnost určit, zda vyhledávání bude probíhat mezi projekty nebo mezi technickými zprávami. Pro usnadnění vyhledání určitého projektu se při vysoké relevanci dokumentu k hledanému výrazu zobrazí jeho náhled. Uživatel pak je schopen získat o něm informace bez toho, aby musel přejít na stránku daného projektu.

3.2 Extraktor dat

Extraktor dat použitý u vytvářeného portálu prochází určené stránky a provádí již zmíněný web scraping. Získávání dat probíhá z velké části pouze na webovém portálu Cordis, a dále

pak z oficiálních stránek o jednotlivých projektech. Systém postupného procházení odkazovaných URL zůstává, je však omezen tak, aby neprocházel celý portál, ale pouze stránky, kde jsou potřebné informace.

Na vstup extraktoru se předá URL portálu Cordis s výpisem projektů. Ve výpisu jsou postupně nalezeny URL jednotlivých projektů a ukládány do textového souboru. Po získání odkazů na všechny projekty začne extraktor postupně stránky projektů otevírat, extrahovat a ukládat data týkající se projektu. Pomocí regulárních výrazů se pokusí najít akronym projektu, celý název projektu, pod jaký program a podprogram se řadí, číslo projektu, oficiální web, kdy začal/začne, kdy skončí/skončil, kdy byl naposledy aktualizován, jaké jsou náklady, jakým způsobem byl financován, výši příspěvku od EU, v jakém státě a jakou univerzitou byl projekt koordinován a kontaktní údaje na koordinátora, jednotlivé účastníky projektu a text celkového cíle projektu. Další krok extrakce je proveden, když se podaří nalézt technické zprávy vydané v rámci projektu, přístupné prostřednictvím portálu Cordis. Dojde k jejich stažení, normalizaci a uložení společně s odkazem na dokument a oficiálním názvem dokumentu. Po extrakci všech projektů a jejich technických zpráv extraktor nalezne ty projekty, které měly uvedenou oficiální stránku projektů. Pomocí části předešlé bakalářské práce [15] se pokusí najít technické zprávy i na nich. Extraktor na závěr provede analýzu, kolik projektů a dokumentů bylo nalezeno a kolik se jich skutečně podařilo extrahovat.

Extraktor primárně sloužil jako nástroj k získání dostatečného množství demonstračních dat, ale je využíván i k pravidelné aktualizaci dat. Získávání aktuálních dat probíhá opět z webového portálu Cordis. K zjištění, které projekty byly přidány nebo aktualizovány, se využívá kontrola data poslední aktualizace, která se nachází u každého projektu. Aktualizace pak může probíhat pravidelně každý týden pomocí softwarového démona Cron.

3.2.1 Využité knihovny

Pro implementaci extraktoru dat byl zvolen skriptovací jazyk Python, který nabízel nejefektivnější rozhraní při komunikaci s databází Elasticsearch (modul `elasticsearch-py`).

Nejpoužívanější knihovnou v extraktoru dat byl modul `re`, tedy knihovna určená pro práci s regulárními výrazy. Ty byly využity tak, aby přesně definovaly místo HTML kódu, odkud se má provést extrakce dat. Jelikož základní extrakce probíhala jen na portálu Cordis, který má strukturu informací o projektech až na pár odlišností stejnou, mohly se regulární výrazy využívat opakovaně. Zde je příklad HTML kódu, kde se nachází datum počátku a konce projektu:

```
<div class="projdates">
<b>From</b> 2009-11-01 <b>to</b> 2011-12-31</div>
```

Zde je regulární výraz, který je schopný datum z HTML kódu získat:

```
<div class="projdates[^"]*">\s<b>From</b>\s?([0-9-]+)\s?
    <b>to</b>\s?([0-9-]+)\s?
```

S ohledem na fakt, že extraktor pracuje s daty z obecně nespolehlivých zdrojů (webových služeb), která jsou přenášena přes datový kanál s nedefinovanou spolehlivostí přenosu, je nutné v průběhu extrakce správně reagovat na možné chyby. Z tohoto důvodu byly v knihovně `common.py` implementovány funkce zajišťující stažení zdroje specifikovaného

pomocí adresy URL, včetně případného opakování přenosu v případě chyby. K implementaci této funkcionality je použita knihovna `urllib2` a možnost nastavení časového limitu (`setdefaulttimeout()`) poskytovaná Python modulem `socket`.

Ke konverzi obsahu PDF dokumentu (více popsané v sekci 3.2.4) do podoby prostého textu byla vytvořena funkce v knihovně `common.py` využívající modul `pdfminer` [25]. Uvedený modul je schopen převodu vstupního dokumentu do prostého textu stránku po stránce s plnou podporou znakové sady UTF-8.

3.2.2 Parametrizovatelnost extraktoru

Parametrizace extraktoru byla zavedena kvůli rozšíření možností jeho využití. Pomocí parametru `-u` nebo `--url` se skriptu předává URL portálu Cordis, kde se nachází výpis projektů, které mají být staženy. Pro demonstrační data byly využity projekty spadající pod program FP7. Extraktor není využíván pouze k jednorázovému stahování dat, ale také k pravidelné aktualizaci. K tomu slouží druhý parametr `-r` nebo `--refresh-interval`, pomocí kterého lze zadat dvě data ve formátu DD/MM/YYYY, specifikující rozmezí projektů podle data poslední aktualizace projektu. Při pravidelné aktualizaci může být např. uvažován předešlý týden. Dále se tento parametr využívá z důvodu, že portál Cordis, není schopen zobrazit výpis s více než 5010 projekty. Pomocí tohoto parametru lze rozdělit jednorázovou extrakci na několik menších extrakcí nepřesahující tento počet.

```
python extractor.py -u <URL ze portálu Cordis>
    -r <rozmezí dat ve formátu DD/MM/YYYY>
```

Příklad pro extrakci všech projektů aktualizovaných během ledna roku 2015:

```
python extractor.py
-u  http://cordis.europa.eu/projects/result_en?q=programme/code%3D'FP7'
-r  01/01/2015 31/01/2015
```

Druhý způsob, jak spustit skript, je pomocí parametru `-f` nebo `--file`, ke kterému se přiřadí adresa textového souboru obsahující URL adresy jednotlivých projektů. Tento parametr byl vytvořen pro případ, že dojde k nečekanému ukončení extraktoru dat, ale adresy stránek projektů již byly do textového souboru uloženy. Dále se může využít v případě, je-li nutno přidat jeden či více konkrétních projektů, které nelze určit pomocí jediné URL.

```
python extractor.py -f <textový soubor s URL projektů>
```

Příklad extrakce projektů nacházející se v souboru `projects_url.txt`:

```
python extractor.py -f projects_url.txt
```

Příklad syntaxe textového souboru, který může být předán pomocí parametru `-f`:

```
/project/rcn/97664_en.html
/project/rcn/99777_en.html
/project/rcn/98840_en.html
/project/rcn/99920_en.html
/project/rcn/96420_en.html
```

3.2.3 Nalezení dokumentů na stránkách projektu

V porovnání se stávajícím informačnímu portálem Cordis poskytuje nově vytvářený portál možnost automatického indexování relevantních dokumentů přímo z webových stránek projektů. K vyhledání dokumentů byla použita stávající implementace z předešlých projektů RRSDeliverables [15, 19, 6]. Cílem projektů bylo vytvoření nástroje pro vyhledávání výzkumných zpráv z webových stránek projektů. Z implementace jsou využity části zajišťující dohledání stránky, na které se technické zprávy vyskytují, vyhledání dokumentů na dohledané stránce a zjištění relevantních informací o technické zprávě (autor, vydání, datum, popis). Veškeré výše uvedené informace nástroj ukládá ve formě souborů XML. Stažení výzkumných zpráv je pak provedeno s využitím dříve popsaných funkcí v knihovně `common.py`.

3.2.4 Extrakce obsahu z PDF a jeho normalizace

Před uložením prostého textu technické zprávy do databáze je nutné provést jeho extrakci a normalizaci. Ve své práci uvažuji pouze možnost extrakce z formátu PDF (*portable document format*), který je v oblasti výzkumných zpráv nepsaným standardem.

Po získání obsahu dokumentu je třeba provést jeho částečnou normalizaci, která si klade za cíl zejména odstranění částí textu zbytečně zhoršujících kvalitu vyhledávaní. Normalizace je prováděna s využitím níže uvedených regulárních výrazů:

- odstranění dlouhých posloupností pomlk a teček:
  "`\.\s*\.\s*\.\s*[\.\s*]+`" a "`-\s*-\s*-\s*[-\s*]+`"
- odstranění *escape* sekvencí značících bílé znaky: "`\[ntr]`"
- zrušení přebytečných bílých znaků: "`\s+`" se přepíše na mezeru,
- spojení rozdělených slov: "`(\w)- (\w)`" je přepsáno pomocí substituce jako "`\1\2`", tj. je vynechána dvojice znaků "`- `".

Pokročilá normalizace zahrnující (např. stemming) není v případě použití NoSQL databází nutná, jelikož bývá těmito databázemi podporována interně.

3.3 Uložení extrahovaných dat

Jak již bylo zmíněno, pro ukládání dat byla vybrána databáze Elasticsearch. NoSQL databáze zaměřená na ukládání dokumentů a plnotextové vyhledávání.

3.3.1 Struktura databáze

V databázi Elasticsearch se data neukládají do tabulek jako v relačních databázích, nýbrž do *indexů*, které jsou unikátně pojmenovány. Navrhovaný webový portál má data rozdělena do dvou indexů. Jeden je využívaný k vyhledávání mezi projekty a jeden pro vyhledávání mezi technickými zprávami. Zde muselo být rozhodnuto, zda ušetřit místo v paměti a každá zpráva se bude odkazovat na projekt ke kterému náleží nebo bude užita určitá míra redundance a informace o projektech se budou u jednotlivých technických zpráv opakovat a rychlost vyhledávání a následovného výpisu se značně urychlí. Rychlost vyhledávání byla shledána jako prioritnější a indexy byly navrženy podle druhé varianty.

Data se přidávají do Elasticsearch indexů po množinách datových polí označovaných jako *dokumenty*. Při vložení prvního dokumentu dojde k vytvoření záznamu, jehož pole jsou z pohledu dalšího zpracování textu zpracována výchozím způsobem. V tomto výchozím režimu jsou hodnoty jednotlivých polí *tokenizovány* (tj. členěny na lexikální jednotky) po jednotlivých slovech. Toto rozčlenění je vhodné pro plnotextové vyhledávání, ale nepoužitelné u dříve zmíněných fasetů. Fasety na základě vybraného pole sjednotí do seznamu hodnoty, které se v daných polích jednotlivých dokumentů nachází. V případě fasetů je tedy nutné chápat hodnoty jako slovní spojení tak, aby např. název státu "United Kingdom" nebyl chápán jako oddělené hodnoty "United" a "Kingdom".

Kromě tokenizace textu je při vyhledávání nutné zohledňovat i fakt, že určitá hledaná lexikální jednotka může být zapsána několika ekvivalentními způsoby (např. různým použitím velkých a malých písmen nebo synonym). Z tohoto důvodu disponuje Elasticsearch databáze možností aplikovat *filtr* na již tokenizovaný text umožňující jeho dodatečnou změnu. Kombinace určitého tokenizátoru a filtru je pak označována jako *analyzátor*. Určení způsobu analýzy jednotlivých polí je v Elasticsearch prováděno pomocí tzv. *mapování*, které přiřazuje každému poli jeho analyzátor. Níže je naznačen příklad analyzátoru použitého pro pole s názvem státu:

```
'analyzer': {
    'analyzer_keyword': {
        'tokenizer': 'keyword',
        'filter': 'lowercase'
    }
}
```

Výše uvedený analyzátor značí, že při porovnání má být hodnota chápana jako slovní spojení porovnávané za použití malých písmen.

3.3.2 Využité knihovny

Jak extraktor dat, tak webový portál je napsán ve skriptovacím jazyku Python. Bylo zapotřebí najít knihovnu, která umožňuje komunikaci s Elasticsearch databází právě pomocí něj. Jako nejvhodnějších se jevil oficiální Elasticsearch klient `elastisearch-py`, který je schopný téměř všech operací potřebných pro práci s databází. Nabízí velmi jednoduché připojení, vkládání záznamů a základní vyhledávání.

Při tvorbě složitějších příkazů pro vyhledávání v Elasticsearch databázi se mi osvědčila knihovna `elasticutils`. Tato knihovna nabízí velmi elegantní přístup k často využívaným funkcionalitám databáze. Vyhledávání pomocí této knihovny se označuje jako *líné* (angl. lazy). Při zadávání požadavků na vyhledání, knihovna pouze sestavuje interní dotaz, ale k databázi přistoupí až v okamžiku, kdy je to skutečně nutné. Také jako velmi praktická vlastnost se ukázala možnost kombinování filtrů, využívá se, když uživatel postupně přidává specifikace na vyžadovaná data.

3.3.3 Efektivní konstrukce databázových dotazů

Existuje několik způsobů, jak se dotazovat na Elasticsearch databázi. Příklady, na kterých budou jednotlivé způsoby vysvětlovány, pochází ze skutečné implementace portálu. Při vytvoření spojení s databází Elasticsearch dojde za pomoci knihovny `elasticutils` ke vzniku proměnné `basic_s`. Proměnná `basic_s` zpočátku uchovává dotaz specifikující index

a všechny jeho záznamy. Tento dotaz se může konkretizovat pomocí vyhledávání či filtrů, a poté jej uložit do jiné proměnné, v našem případě `query_s`. Tu je potřeba omezit tak, aby výsledná proměnná obsahovala dotaz pouze na uživatelem žádané záznamy. Mezi operace pro nalezení hledaných dokumentů patří *query*. V mém případě využívám *query_raw*, která nám dává větší množství možných specifikací:

```
query_s = basic_s.query_raw({
    "multi_match" : {
        "query" : keyword,
        "fields" : [ "abbr^6","title^5", "subprogramme^3", "objective",
            "origWeb"],
    }
})
```

Tato query je využívána při vyhledávání zadaného výrazu, který se nachází v proměnné `keyword`. Specifikaci `multi_match` je použita kvůli potřebě nalezení výrazu ve více než jednom poli. Všechna prohledávaná pole jsou pak vyjmenována ve „fields“ společně s číslem určujícím jejich důležitost při vyhledávání.

```
query_s = query_s[from:to]
```

Tato další specifikace je použita za účelem stránkování projektů. Podle toho, na kterou stránku uživatel vstupuje, se spočítá, od kolikátého projektu po kolikátý se budou zobrazovat. Jelikož se `elasticutils` využívá již zmíněné líné vyhodnocení (angl. lazy evaluation), zatím stále nedošlo k žádnému přistoupení do databáze. Pouze se tato nová specifikace přidala do dotazu `query_s`.

```
query_s = query_s.highlight('objective', pre_tags = ["<b>"]
post_tags = ["</b>"])
```

Další praktickou vlastností rozhraní Elasticsearch je možnost jednoduchého vytváření textových náhledů. Definováním `pre_tags` a `post_tags` HTML značkou `<b>` způsobí, že hledaný výraz v textovém náhledu bude zobrazen tučně.

Pro vytváření menu byla využita fasetová klasifikace, a pak následovně na základě zvolení položky menu uživatelem databáze filtrována a vypsána:

```
facet_s = basic_s.facet("coord", filtered=True)
```

Fasety se vytvářejí na základě určeného pole, konkrétně zde se provádí nad polem koordinátora projektu. V proměnné `facet_s` se pak nachází seznam s unikátními hodnotami, které dané pole obsahují, seřazené podle počtu výskytu.

```
query_s = basic_s.filter(**search_dic)
```

Potom, co si uživatel jednu z položek z menu vybral, je hodnota dané položky vložena do JSON struktury `search_dic`. Databáze je pak na základě daného JSON filtrována a v `query_s` už je odkaz jen na specifikované položky.

Pro vyhledávání souvislostí mezi projekty a získání projektů navzájem podobných se využívá funkce `more-like-this` (`MLT`). Ta je schopna na základě vybraného dokumentu získat odkaz na seznam podobných seřazených podle relevance.

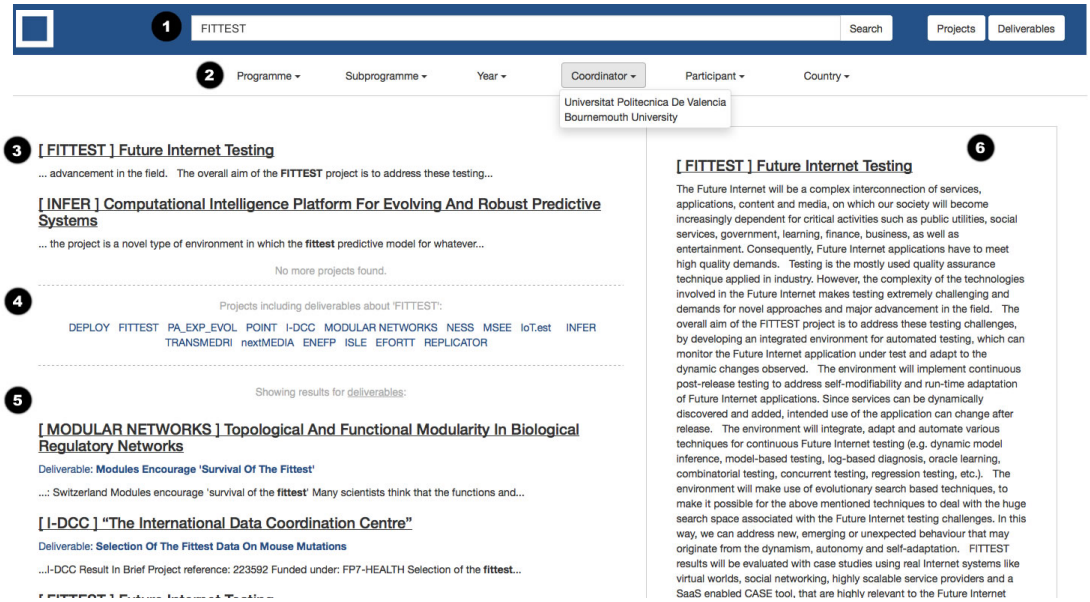
```
mlt_s = MLT(projectid, index=INDEX_PROJ, doctype=DOCTYPE)
```

3.4 Webový portál

Struktura portálu je založena na jednoduchém a přehledném ovládání s možností konkrétněji specifikovat požadovaný výsledek. Umístění jednotlivých prvků nápadně připomíná webový vyhledávač Google, a to z důvodů, aby se nový uživatel jednoduše zorientoval.

3.4.1 Zobrazovací možnosti portálu

Webový portál má tři různé struktury stránek. Úvodní strana, strana zobrazující výsledek vyhledávání a strana zobrazující informace o konkrétním projektu. Úvodní strana je velmi přímočará nabízí uživateli pouze možnost zadat hledaný projekt nebo výraz, který má být v projektu obsažen. Po zadání výrazu, v našem případě je výraz akronym projektu „FITTEST“, se vygeneruje stránka na obrázku 3.2.



Obrázek 3.2: Podoba strany s výpisem výsledků vyhledávání.

1. Vyhledávací pole se přemístilo na horní část obrazovky a přibyla mu možnost výběru, zda má vyhledávání probíhat pouze na základě informací o projektu nebo vyhledávat v dokumentech projektů.

2. Menu vytvořené pomocí roletkových seznamů umožňuje specifikovat hledané projekty na základě programu, podprogramu, roku, koordinátora, zúčastněných a země. Pro pokročilé uživatele není nutné toto menu využívat a specifikace může zapsat přímo do hledaného pole, viz obrázek 3.3.

Obrázek 3.3: Detail dotazovacího pole webového portálu.

3. Jsou-li nalezeny projekty obsahující daný výraz, dojde nejprve k výpisu právě jich. Ve hranatých závorkách se nachází akronym projektu a následuje název projektu.

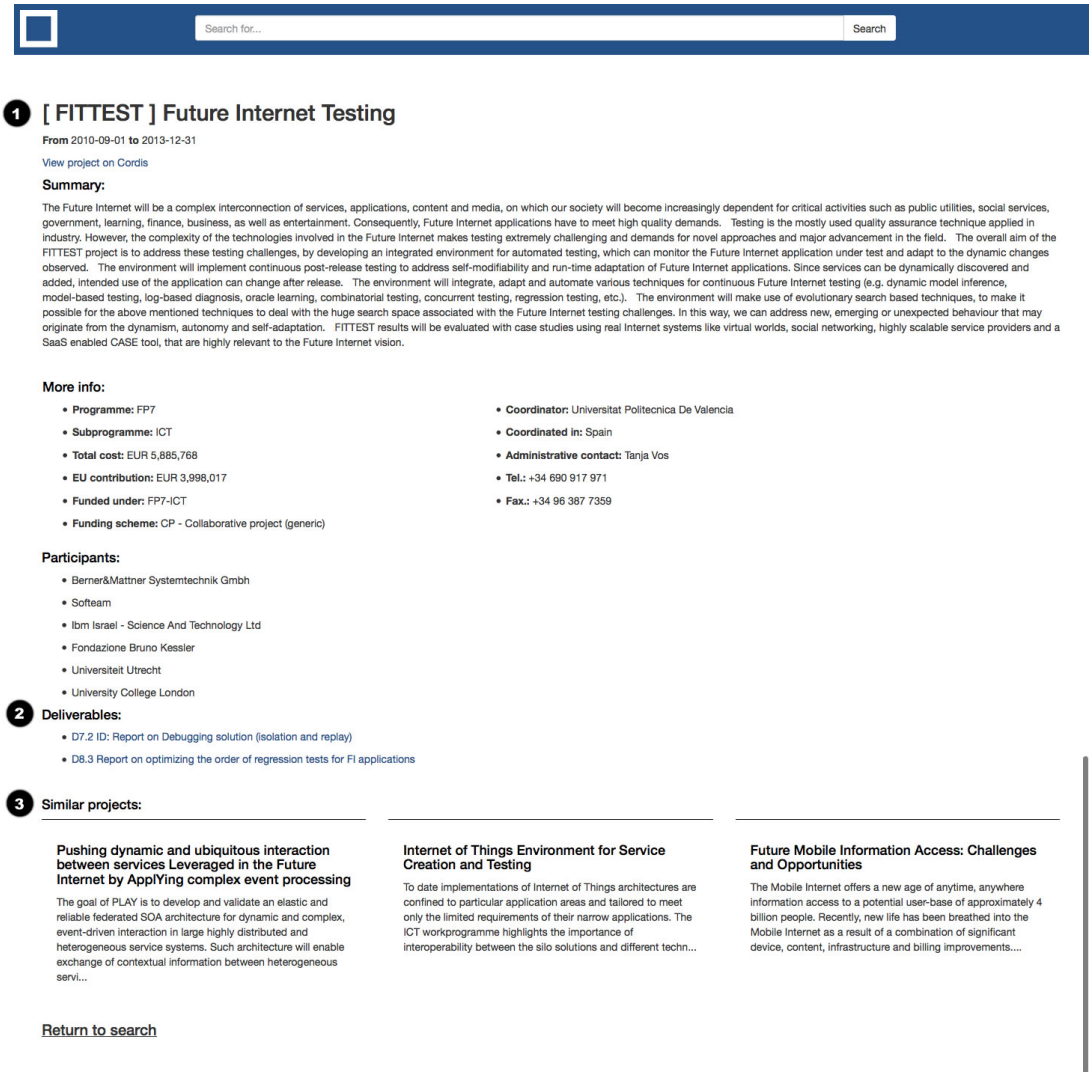
Jestliže byl hledaný výraz nalezen v popisu projektu, zobrazí se část textu, kde se nacházel a je zvýrazněn tučným písmem.

4. Jestliže byl na základě požadavků nalezen malý počet projektů, tj. méně než 20, portál nabídne uživateli projekty mající dokumenty, které hledaný výraz obsahují.

5. Následně nabídne ty nejrelevantnější dokumenty. Výpis se skládá z akronymu projektu, názvu projektu a přímého odkazu na PDF dokument. Pomocí ukázky textu znázorňuje kontext, v kterém byl výraz v dokumentu použit.

6. Náhled na projekt se zobrazí, když byl konkrétně vyhledán akronym projektu, jako je tomu v tomto případě nebo když relevance projektu k hledanému výrazu je natolik velká, že skóre daného projektu přesáhne určenou hranici.

Po výběru požadovaného projektu se vygeneruje stránka na obrázku 3.4.



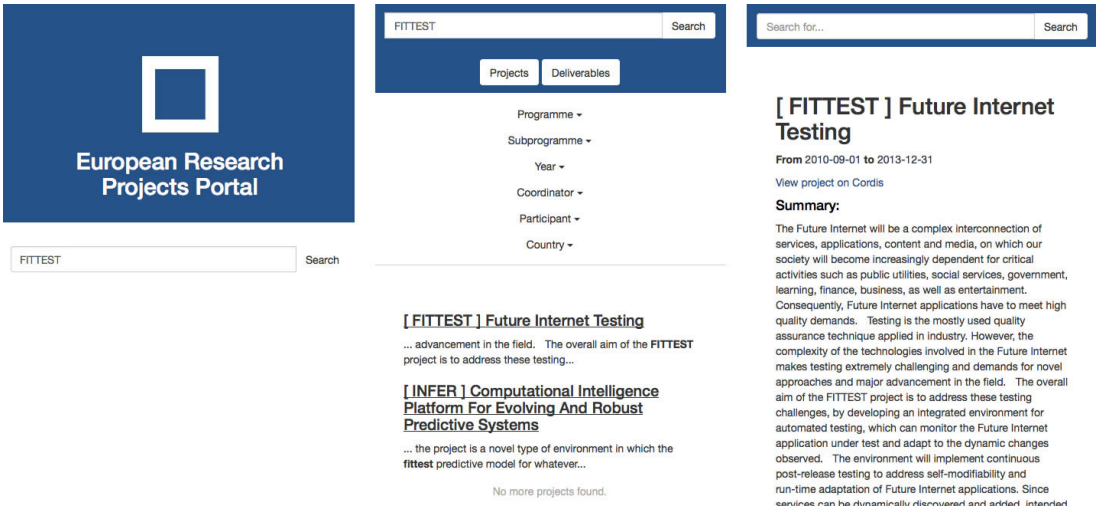
Obrázek 3.4: Podoba strany s výpisem informací o projektu.

1. Nejprve se zobrazují veškeré informace získané o daném projektu.

2. Pod nimi se nachází seznam technických zpráv, které jsou k danému projektu přiřazeny.

3. Na závěr jsou vybrány tři jiné projekty, které byly vybrány na základě největší relevance k danému projektu pomocí funkce more-like-this, a mohly by tedy uživatele zajímat.

Jelikož byl webový portál vytvořen pomocí rozhraní Bootstrap, je velikost a rozložení jednotlivých prvků na stránce dynamické. Portál se tedy bude zobrazovat správně na většině běžných zařízení jako je tablet nebo chytrý telefon, viz obrázek 3.5.



Obrázek 3.5: Ukázka vzhledu webového portálu na chytrém telefonu.

3.4.2 Gramatika použitá pro vyhledávací pole

Vyhledávací pole je primárním prvkem sloužícím pro zadávání uživatelských dotazů pro webový portál. V souladu s dříve popsanými principy tvorby uživatelských rozhraní, které kladou důraz na jednoduchost a přehlednost, umožňuje vyhledávací pole zadávání rychlých dotazů prostým zapsáním klíčových slov. Pokročilé dotazování je pak možné provádět pomocí tzv. *specifikací*, které slouží k zpřesnění výsledků vyhledávání.

Pro zpracování dotazovacího jazyka přijímaného vyhledávacím polem byla navržena bezkontextová gramatika vycházející z formátu JSON. Vstupním bodem gramatiky jazyka níže zapsané pomocí rozšířené Backus-Naurovy formy je neterminál `<query>`, který se dále rozvíjí na jednotlivé části dotazu reprezentované neterminálem `<qelem>`. Jednotlivé elementy jsou odděleny neprázdnou sekvencí bílých znaků (v gramatice označených terminálem `ELEM_SEP`). Zpracování elementů dotazu je nezávislé na jejich pořadí.

```
<query> ::= <qelem>
          | <query> ELEM_SEP <qelem>
```

Elementy `<qelem>` jsou pak buď klíčová slova zapsaná ve formě textového řetězce `<string>`, nebo speciálně skládající se z dvojice řetězů oddělených znakem dvojtečka, kde první řetězec značí název filtru a druhý filtrovanou hodnotu. Povolenými názvy filtru jsou hodnoty: „country“ (stát hlavního řešitele), „programme“ (program), „subprogramme“ (podprogram), „coordinator“ (hlavní řešitel), „participant“ (řešitel) a „year“ (rok započetí). Specifikace je

možné do vyhledávacího pole zapisovat přímo, nebo lze využít připravených roletkových menu dostupných z panelů nacházejících se pod vyhledávacím polem. Při zapsání více než jedné specifikace je implicitně uvažována jejich konjunkce.

```
<qelem> ::= <string>                # klíčové slovo
          | <string> ':' <string>   # specifikace
```

Řetězce `<string>` používané v klíčových slovech a specifikacích je možno zadat přímo, nebo uzavřené mezi znaky uvozovka. Druhý způsob je nutný v případě, chceme-li uvést specifikace obsahující hodnotu skládající se z více slov.

```
<string> ::= <chars>
           | '"' <chars> '"'

<chars> ::= <char>
          | <chars> <char>
```

Mezi znaky `<char>` řetězců `<string>` jsou také povoleny tzv. *escape sekvence* umožňující zápis znaků jinak použitých pro formování dotazu.

```
<char> ::= UNESCAPED    # vše kromě '"', '\', ':'
         | '\' '"'
         | '\' '\'
         | '\' ':'
```

K implementaci syntaktického a sémantického analyzátoru pro zpracování dotazovacího jazyka byl zvolen nástroj PLY [5]. PLY implementuje standardně používané programy LEX a YACC pro tvorbu překladačů v prostředí Pythonu. PLY dovoluje přepis gramatiky navržené v Backus-Naurově formě do programového kódu v podobě LALR gramatiky s tím, že každému přechodu gramatiky umožňuje přiřadit jeho sémantiku. V tomto případě je při zpracování dotazu postupně tvořena struktura obsahující seznam klíčových slov a seznam specifikací. Zároveň je také uživatelský vstup kontrolován na (ne)přítomnost zakázaných znaků. Výše uvedená funkcionalita je implementována v modulu `query.py`.

3.4.3 Skóre a řazení projektů

Teoretický úvod výpočtu skóre byl představen v podkapitole 2.2.2. Víme tedy, že skóre určuje relevanci k vyhledávání a podle něj jsou jednotlivé dokumenty řazeny. Má tedy přímý vliv, v jakém pořadí bude uživateli výpis záznamů zobrazen.

Výsledné pořadí nemusí čistě záležet na interních výpočtech Elasticsearch databáze, ale může být upravené pomocí dotazu:

```
"fields" : [ "abbr^6", "title^5", "subprogramme^3", "objective"]
```

Tato část dotazu, která určuje, v kterých polích se má zadaný výraz hledat. Čísla u názvů polí tzv. *váhový vektor* nám násobí podstatnost nalezení výrazu v jednotlivých polích a tím má vliv na výsledné skóre. Nalezení výrazu v nadpisu má pětkrát větší váhu jak nalezení ve výrazu popisu projektu (objective). Dotaz je uzpůsoben tak, že s velkou pravděpodobností se prvně vypíší projekty, kde hledaný termín je jako akronym projektu nebo je obsažen

v názvu. Projekty, které hledaný termín obsahují pouze jako zmínku v popisu projektu, budou řazeny až na úplný konec.

Skóre bylo vyžito na nově navrhovaném portálu ještě jednou, a to při určování projektu pro zobrazení v rámci okamžitého náhledu nacházející se na pravé straně výpisu projektů či zpráv. Když skóre záznamu překročí určitou hodnotu, nad kterou je velmi pravděpodobné, že hledán je právě onen projekt, zobrazí se do okamžitého náhledu.

3.4.4 Využitá rozhraní

Webový portál byl vytvořen za pomocí webového aplikačního rozhraní Flask [23], který je napsán ve skriptovacím jazyce Python. Flask je označován jako mikrorozhraní z důvodu, že je jeho jádro velmi jednoduché, ale rozšiřitelné. Umožňuje uživateli přidávat pouze ty knihovny a nástroje, které potřebuje.

Externí knihovna, kterou jsem využila k rozhraní Flask, se nazývá Jinja2 [24]. Je to šablonovací jazyk pro Python, který nabízí velmi přehledný způsob propojení funkčního skriptu a šablon. Šablony jsou kódy psané ve značkovacím jazyce HTML a pomocí právě Jinja2 do nich lze dynamicky vpisovat hodnoty ze skriptu, který běží na pozadí. Několik příkladů použití Jinja2:

Příklad vepsání proměnné do HTML.

```
<h1> {{ title }} </h1>
```

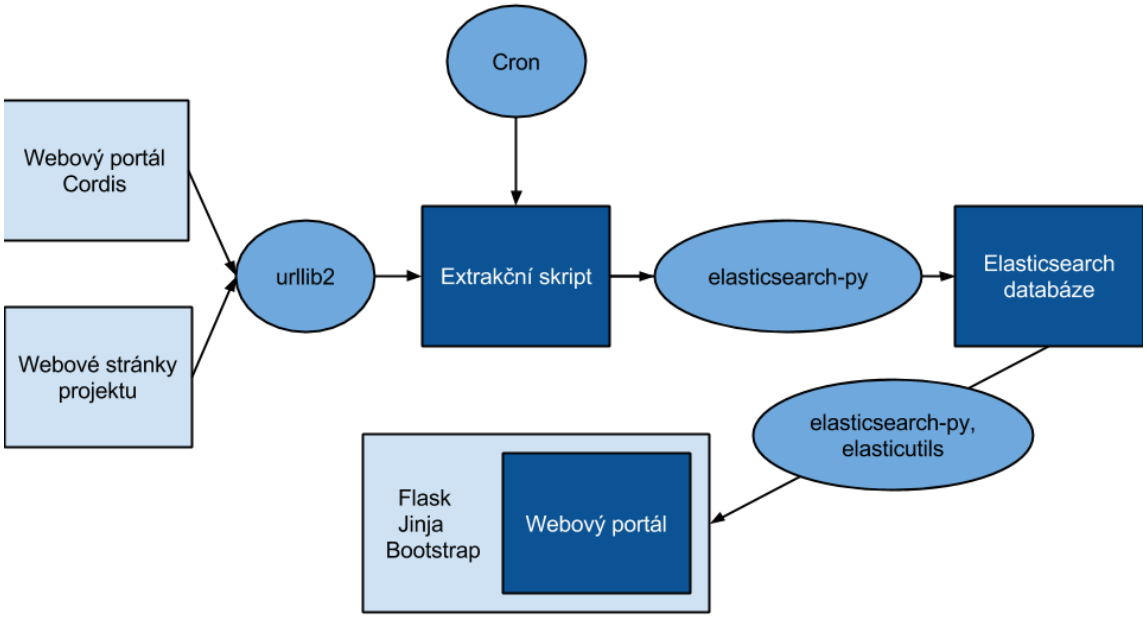
Iterace seznamem a kladení podmínek v HTML.

```
{% if list %} <!-- jestliže seznam list existuje -->
<ul>
    {% for item in list %} <!-- procházej prvek po prvku -->
        <li> {{ item }} </li>
    {% endfor %}
</ul>
{% endif %}
```

Další z rozhraní je Bootstrap [3], který je široce využíván pro vytváření webových aplikací. Při použití tohoto rozhraní je pak jeden kód ve značkovacím jazyce přehledně zobrazitelný na všech běžných zařízeních, od běžného monitoru po chytré telefony. Svou oblibu si získal i pro svůj čistý a moderní vzhled prvků jako jsou tlačítka, menu a další.

3.5 Komunikace mezi jednotlivými částmi

Jelikož na sebe jednotlivé části implementace navazují, musí být od počátku definován způsob, jak mezi sebou komunikují. Graf na obrázku 3.6 zjednodušeně znázorňuje, kudy a pomocí kterých knihoven jsou data předávána mezi jednotlivými částmi projektu. Pomocí knihovny `urllib2` získá extraktor dat na základě URL zdrojové kódy stránek webového portálu Cordis nebo z webových stránek jednotlivých projektů. Pomocí softwarového démona Cronu pak probíhá pravidelná automatická extrakce. Extraktor se připojuje k Elasticsearch pomocí knihovny `elasticsearch-py` a pomocí ní do databáze vkládá data. Přes stejnou knihovnu a za použití další knihovny `elasticutils` dojde k propagaci dat z Elasticsearch databáze do webového portálu. Frameworky Flask, Jinja a Bootstrap pak slouží ke generování požadované stránky uživateli.



Obrázek 3.6: Graf reprezentující komunikaci mezi jednotlivými částmi projektu.

Kapitola 4

Testování portálu a extraktoru dat

Správnost implementace jednotlivých částí byla ověřena dvěma typy testů, které více popisují následující podkapitoly. První představuje způsob realizace *jednotkových testů* (angl. unit test). Druhá sekce se pak zaměřuje na použité *integrační testy*, ověřující projekt jako celek. Testování probíhalo v prostředí operačního systému Linux (distribuce Ubuntu 14.04, 32bit).

4.1 Jednotkové testy

Jednotkové testy byly v průběhu implementace vytvářeny ke každému modulu. Pro tento typ testů byl použit testovací systém interpretu jazyka Python nazývaný `unittest`. Při použití nástroje je kromě testu samotného také třeba definovat funkce, které mají být zavolány před/po zpracování daného testu—funkce `setUp()` a `tearDown()`. Implementace testu pak k ověření požadované funkcionality zpravidla využívá funkci `assert()` a další od ní odvozené. Příklad jednotkového testu pro modul `query.py` je uveden níže:

```
class TestQuery(unittest.TestCase):
    def test_ComplexQueries(self):
        q = Query('nuclear programme:fp7 country:"Czech Republic"
            "subprogramme":PEOPLE')
        self.assertTrue("nuclear" in q.keywords)
        self.assertEqual(q.specifications["programme"], "fp7")
        self.assertEqual(q.specifications["country"], "Czech Republic")
        self.assertEqual(q.specifications["subprogramme"], "PEOPLE")
```

4.2 Integrační testy

Integrační testy slouží k ověření splnění specifikačních požadavků u projektu jako celku. Dohromady testují dříve popsané položky, které byly předmětem návrhu a následné implementace, testována je tedy zejména komunikace mezi jednotlivými spolupracujícími skupinami prvků.

Pro potřeby této práce byla manuálně vybrána skupina 20 projektů, které mají vlastní webové stránky. Informace a technické zprávy těchto projektů byly posléze zpracovány extraktorem dat a uloženy do databáze. Následně pak bylo využito vyhledávání poskytované portálem pro ověření extrahovaných informací. Úspěšnost integračních testů je shrnuta v tabulce 4.1.

Tabulka 4.1: Úspěšnost integračních testů.

Testovaná položka	Skutečná hodnota	Správně extrahováno	Úspěšnost [%]
Akronym projektu	20	20	100%
Celý název	20	20	100%
Program	20	20	100%
Podprogram	20	20	100%
Číslo projektu	20	20	100%
Oficiální web	9	7	77,8%
Datum začátku	20	20	100%
Datum konce	20	20	100%
Náklady	20	20	100%
Způsob financování	20	20	100%
Příspěvek EU	20	20	100%
Koordinátor	20	20	100%
Účastníci	18	18	100%
Cíl projektu	20	13	65%

Většina dat získávaných na webovém portálu Cordis měla stálou strukturu a extraktor dat měl 100% úspěšnost při jejich získávání. Portál Cordis umožňuje uvádět oficiální web v položce „Multimedia“, která ovšem není pro něj přímo vyhrazená. V několika případech se zde vyskytují i jiná data, vedoucí k částečné neúspěšnosti získání této položky. Cíl projektu nemá přesně vyhraněné místo ve struktuře HTML stránky a i přes několik způsobů vyhledávání se občas nepodaří jej extrahovat. Počet získaných dokumentů je pak závislý na faktu, zda se podařilo soubor otevřít a extrahovat. Jestliže se nepodařilo, nedojde k jeho uložení.

Kapitola 5

Závěr

Cílem práce bylo navrhnout a implementovat informační systém publikující výsledky evropských výzkumných projektů s možností snadného získávání přehledů a pokročilého dotazování. Výsledkem práce je automaticky aktualizovaný webový portál pracující nad databází obsahující přes 21 000 evropských výzkumných projektů a přes 30 000 technických zpráv. Z 96% byly tyto zprávy staženy z evropského vědeckého portálu Cordis. Zbylá 4% pak představují zprávy stažené z oficiálních stránek jednotlivých projektů, které do systému Cordis nebyly zadány. K nalezení těchto dokumentů byla využitá část implementace z dříve vypracovaných projektů RRSDeliverables vyvíjeného skupinou ReReSearch na FIT VUT v Brně. Na rozdíl od portálu Cordis navíc nově implementovaný portál umožňuje v technických zprávách i plnotextové vyhledávání. Plně automatizovaný extraktor dat, který je součástí implementace portálu, každý měsíc nalézá a přidává do databáze v průměru dalších 20 projektů.

Do budoucna je portál dále možné rozšířit např. o možnost extrakce dat i z jiných portálů nebo možnosti přidávání projektů přímo na webový portál. Dalším příkladem rozšíření může také být modifikace gramatiky dotazovacího pole s cílem poskytnout uživateli ještě větší možnosti při specifikaci hledaného projektu či dokumentu.

Z důvodu používání pro mě nových rozhraní a technologií, např. Elasticsearch, Flask, Jinja, bylo komplikované přesně naplánovat a předvídat všechny problémy, které mohou nastat. Práce na jednotlivých částech vědeckého portálu taktéž vyžadovala u zmíněných technologií hluboké pochopení jejich principů a funkcí. Jsem spokojená s výsledným stavem své bakalářské práce. V nynější fázi se jedná o plně funkční webový portál schopný automatické aktualizace nabízející uživateli všechny požadované vlastnosti dané jejím charakterem a zadáním.

Literatura

[1] NoSQL Databases [online]. Dostupné z `http://nosql-database.org`, 2015 [cit. 2015-04-13].

[2] Stránka projektu Apache Lucene [online]. Dostupné z `https://lucene.apache.org/`, 2015 [cit. 2015-04-13].

[3] Responsive Front-End Framework (Bootstrap) [online]. Dostupné z `http://getbootstrap.com`, 2015 [cit. 2015-04-20].

[4] Aho, A. V.; Ullman, J. D.: *Foundations of Computer Science*, kapitola Patterns, Automata, and Regular Expressions. 1992.

[5] Beazley, D.: PLY (Python Lex-Yacc) [online]. Dostupné z `http://www.dabeaz.com/ply/`, 2015 [cit. 2015-03-13].

[6] Beneš, K.: Vyhledávání a zpracování výzkumných zpráv projektů [online]. Dostupné z `https://knot.fit.vutbr.cz/wiki/index.php/Rrs_deliverables3`, 2012 [cit. 2015-04-05].

[7] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format [online]. Dostupné z `http://tools.ietf.org/html/rfc7159`, 2014 [cit. 2015-03-29].

[8] Elasticsearch: Filters [online]. Dostupné z `http://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-filters.html`, 2015 [cit. 2015-02-09].

[9] Elasticsearch: Highlighting Our Searches [online]. Dostupné z `http://www.elastic.co/guide/en/elasticsearch/guide/current/highlighting-intro.html`, 2015 [cit. 2015-02-09].

[10] Elasticsearch: Practical Considerations [online]. Dostupné z `http://www.elastic.co/guide/en/elasticsearch/guide/master/parent-child-performance.html`, 2015 [cit. 2015-02-09].

[11] Elasticsearch: Queries [online]. Dostupné z `http://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-queries.html`, 2015 [cit. 2015-02-09].

[12] Elasticsearch: Terms Facet [online]. Dostupné z `http://www.elastic.co/guide/en/elasticsearch/reference/current/search-facets-terms-facet.html`, 2015 [cit. 2015-02-09].

[13] Elasticsearch: More Like This Query [online]. Dostupné z http://www.elastic.co/guide/en/elasticsearch/reference/1.4/query-dsl-mlt-query.html#query-dsl-mlt-query, 2015 [cit. 2015-04-05].

[14] Elasticsearch: Distributed Search and Analytics [online]. Dostupné z https://www.elastic.co/products/elasticsearch, 2015 [cit. 2015-04-12].

[15] Heller, S.: *Knihovna pro podporu vývoje systému ReReSearch*. Bakalářská práce, FIT VUT v Brně, 2011.

[16] Katsov, I.: NoSQL Data Modeling Techniques [online]. Dostupné z https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/, 2012 [cit. 2015-03-15].

[17] Kehoe, M.: Contrasting Relational Databases and Full-Text Search Engines. *NIE Newsletter*, ročník 2, č. 1, 2004.

[18] Krug, S.: *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. New Riders, 2000, ISBN 9780321965516.

[19] Macko, L.; Skácel, J.; Koroncziová, D.; aj.: Převod informací o deliverables z XML do SQL [online]. Dostupné z https://knot.fit.vutbr.cz/wiki/index.php/Rrs_deliverables2, 2014 [cit. 2015-04-11].

[20] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, 2008, ISBN 9780521865715.

[21] Rajaraman, A.; Ullman, J. D.: *Mining of Massive Datasets*. Cambridge University Press, 2011, ISBN 9781107015357.

[22] Rodriguez-Castro, B.; Glaser, H.; Carr, L. (editoři): *How to Make a Faceted Classification and Put It On the Web*. 2003.

[23] Ronacher, A.: Microframework for Python (Flask) [online]. Dostupné z http://flask.pocoo.org, 2015 [cit. 2015-04-20].

[24] Ronacher, A.: Python Template Engine (Jinja2) [online]. Dostupné z http://jinja.pocoo.org, 2015 [cit. 2015-04-20].

[25] Shinyama, Y.: PDF parser and analyzer [online]. Dostupné z http://pypi.python.org/pypi/pdfminer/, 2014 [cit. 2015-04-20].

[26] Webster, S.: What Is Scraping? The Basics For Everyone [online]. Dostupné z https://myhelpster.com/what-is-scraping-the-basics-for-everyone/, 2015 [cit. 2015-05-07].