České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Sledování matematických článků podle klasifikačních kódů Americké matematické společnosti

Bohdan Frejišyn

Vedoucí práce: Ing. František Štampach Ph.D.

17. května 2016

Poděkování

Tímto bych chtěl poděkovat své rodině, přítelkyni a kolegům z práce za veškerou podporu, kterou mi věnovali. Děkuji také panu Ing. Františku Štampachovi, Ph.D. za konzultace a téma, které mě zaujalo a bavilo zpracovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 17. května 2016 .....

Odkaz na tuto práci

Frejišyn, Bohdan. *Sledování matematických článků podle klasifikačních kódů Americké matematické společnosti*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se věnuje postupu vytváření webové aplikace pro filtrování odborných článků z internetového archivu. Práce představuje řešení, které využívá klasických webových technologií. Zejména byly využity metody pro rozbor elektronických dokumentů a metody pro přehledné zobrazení výsledků. Provedeným výzkumem se zjistilo, že existují omezení ztěžující řešení. Cíl práce byl však přesto splněn. Výsledek této práce umožňují jejímu uživateli sledovat nejnovější články z internetového archivu arXiv.org filtrované podle zadaného dotazu.

Klíčová slova RSS zdroj, MSC kód, arxiv.org, parsování, PDF dokument, filtrování, článek, webová aplikace

Abstract

This thesis solves a process of creating a web application offering filter possibilities for technical articles from an internet archive. Methods of parsing electronic documents so as methods of uncluttered displaying of results were used. By research there was found out that there are restrictions of creating such application. However, the goal of the thesis has been accomplished. The

result of this thesis allows its user to be able to follow most recent articles from internet archive arXiv.org. The articles are specified by user's query.

Keywords RSS feed, MSC code, arxiv.org, parsing, PDF document, filtering, article, web application

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Očekává se od vás, že budete sledovat nejnovější vývoj ve svém oboru, ovšem tento úkol se ukázal být mnohem těžší, než jste si mysleli. Při hledání posledních novinek obdržíte odkazy na 17 milionů stránek, z nichž některé jsou až rok staré. Lze je nějak setřídit, abyste získali novinky jen za poslední měsíc [1]?

S rozvojem internetu přišel i rozvoj webových stránek často publikujících nový obsah, například zpravodajských nebo vědeckých. Vyvstala potřeba úsporně prohlížet takový obsah bez nutnosti několikrát denně navštěvovat dané stránky a znovu hledat pro uživatele relevantní (aktuální) informace.

Úvodní motivační otázka nabízí zamyšlení nad potřebou efektivně prohlížet často publikující webové stránky, které uživatele zajímají, a jak u toho ušetřit co nejvíce času. Jde o zamyšlení v úvodu publikace o RSS technologii, přesněji jazyku nebo standardu pro popis elektronických dat, kterého se práce z velké části dotýká a se kterým pracuje.

Tato bakalářská práce se konkrétně zabývá filtrováním velkého množství článků z webového archivu arXiv.org z oboru matematiky. V práci bude popsána analýza problému, autorův návrh řešení a v neposlední řadě konkrétní implementace webové aplikace zajišťující zmíněnou funkcionalitu. V práci autor mimojiné popisuje technologie, které při tvorbě aplikace využil, rozhodnutí, která ho k použití daných technologií vedla a výsledek, ke kterému autor po analýze a implementaci dospěl.

Hlavními tématy práce jsou:

T1: RSS technologie - verze, možnosti a zdroje.

T2: Archiv článků arXiv.org, jeho komponenty a omezení.

T3: MSC kategorizace a filtrování článků za jejího využití.

Kapitola 1

Cíl práce

Cílů práce bylo vytyčeno hned několik. Jejich sjednocení lze popsat jako webovou aplikaci vykonávající předzpracování původního RSS zdroje do specifičtější podoby.

Jednotlivé aspekty, kterým se autor v rámci práce věnoval, dohromady vytvořily celkový požadavek na výslednou aplikaci. Šlo zejména o:

C1: Schopnost číst RSS zdroj volně získatelný z webu arXiv[1].

C2: Schopnost číst a filtrovat i PDF soubory obsahující zdrojový text publikovaných článků.

C3: Vytvoření specifičtějšího RSS zdroje podle požadavku uživatele.

C4: Možnost v aplikaci zobrazit články neobsahujících zadanou specifikaci.

Výsledný produkt analýzy, návrhu a implementace by měl být dostupný zejména pro pedagogy ČVUT FIT pravidelně navštěvující stránku arXiv.org.

[1] `http://export.arxiv.org/rss/math`

Kapitola 2

Analýza

2.1 MSC klasifikace

Hlavní účel klasifikace článků v literatuře zabývající se matematikou, která používá MSC kódy, je co nejvíce zjednodušit hledání článků v okruhu zájmu čtenáře. Myšlenkou je, že každý takto označený článek příláká zájem co nejvíce potenciálních čtenářů. Velmi užitečnou se jeví situace, kdy autor článku i potenciální čtenář klasifikaci znají a jsou více obeznámeni především s takovou, která je nejblíže jejich zájmům [2]. V tabulce 2.1 autor uvádí část zobecněného přehledu MSC 2010 standardu. Kompletní standard lze získat na stránkách *American Mathematical Society*[2].

2.2 arXiv.org

Co přesně si lze pod pojmem arXiv.org představit?

ArXiv je služba poskytující volný přístup k digitálním vědeckým publikacím v oborech jako například fyzika, matematika nebo informatika. ArXiv je vlastněn a provozován Cornell University, soukromou nevýdělečnou vzdělávací institucí [3].

2.3 Existující řešení

Při analýze autor narazil na dvě hlavní řešení nabízející rozhraní pro vyhledávání:

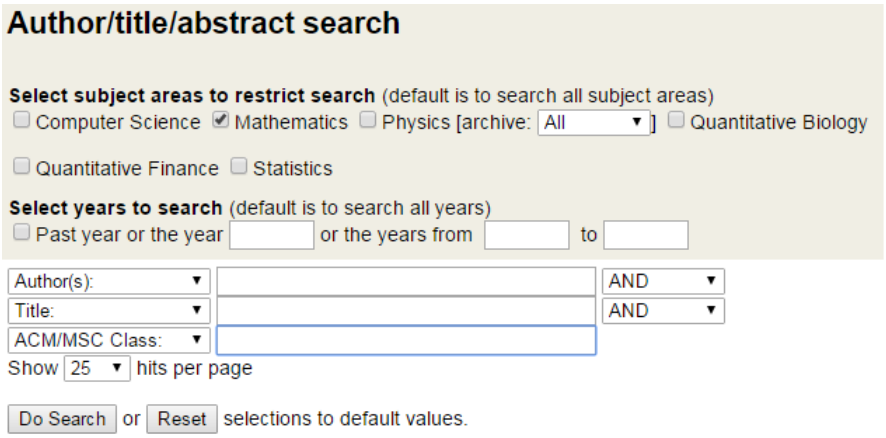
2.3.1 Search arXiv.org

První z řešení, k nahlédnutí na obrázku 2.1, nabízí přímo webová stránka arxiv.org. Frontend tohoto vyhledávače svým vzhledem kopíruje motiv celého

[2] `http://www.ams.org/msc/pdfs/classifications2010.pdf`

Tabulka 2.1: Obecná MSC klasifikace

MSC kód	Okruh témat
01-XX	History and biography [See the classification number –03 in the other sections]
03-XX	Mathematical logic and foundations
05-XX	Combinatorics {For finite fields, see 11Txx}
06-XX	Order, lattices, ordered algebraic structures [See also 18B35]
08-XX	General algebraic systems
11-XX	Number theory
12-XX	Field theory and polynomials
13-XX	Commutative algebra
14-XX	Algebraic geometry
15-XX	Linear and multilinear algebra; matrix theory
16-XX	Associative rings and algebras {For the commutative case, see 13-XX}
17-XX	Nonassociative rings and algebras
18-XX	Category theory; homological algebra {For commutative rings see 13Dxx, for associative rings 16Exx, for groups 20Jxx, for topological groups and related structures 57Txx; see also55Nxx and 55Uxx for algebraic topology}
19-XX	K-theory [See also 16E20, 18F25]
20-XX	Group theory and generalizations
22-XX	Topological groups, Lie groups {For transformation groups, see 54H15, 57Sxx, 58-XX. For abstract harmonic analysis, see 43-XX}
26-XX	Real functions [See also 54C30]
28-XX	Measure and integration {For analysis on manifolds, see 58-XX}
30-XX	Functions of a complex variable {For analysis on manifolds, see 58-XX}
31-XX	Potential theory {For probabilistic potential theory, see 60J45}
32-XX	Several complex variables and analytic spaces {For infinite-dimensional holomorphy, see 46G20, 58B12}
33-XX	Special functions (33-XX deals with the properties of functions as functions) {For orthogonal functions, see 42Cxx; for aspects of combinatorics see 05Axx; for number-theoretic aspects see 11-XX; for representation theory see 22Exx}



Obrázek 2.1: Náhled arxiv.org vyhledávače [5]

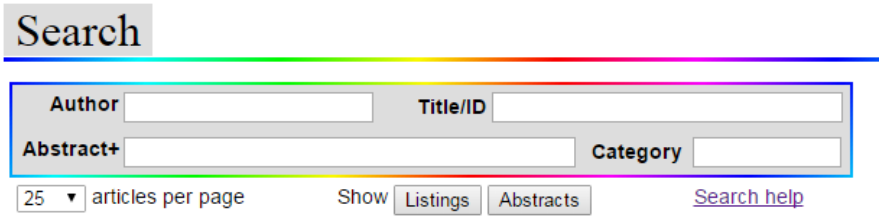
Obrázek 2.2: Náhled arxiv.org chystaného full text vyhledávače [5]

prostředí. Zadání přijímá od uživatele pomocí formuláře obsahujícího několik zaškrtávacích políček a textových polí.

Arxiv.org vyhledávač nabízí uživateli možnost stránkování, což v současnosti představuje jistý standard vzhledem k tomu, že prakticky všechny vyhledávače stránkování využívají [4].

Jako komponentu Search arxiv.org jeho autoři připravili i vyhledávání podle MSC kódů. Toto řešení ovšem pracuje nad celou databází článků a pouze nad jejich metadaty. Narozdíl od přístupu Search arxiv.org autor této práce využívá PDF Parser, tedy PHP knihovnu určenou pro parsing neboli rozbor PDF souborů a extrahování jejich elementů jako text [6].

Nutno podotknout, že vývojáři arxiv.org pracují na full text vyhledávači, který zahrnuje i funkcionalitu vyhledávání MSC kódů uvnitř celého textu publikovaných článků, a tedy i přiložených PDF souborů. Nicméně už z názvu tohoto vyhledávače, viz obrázek 2.2, lze vyčíst, že se jedná, alespoň prozatím, o funkcionalitu stále ve vývoji. Navíc ani tato funkce nefiltruje pouze nejnovější články z aktuálního RSS zdroje a neposkytuje odpovídající výstup v podobě dotazem omezeného RSS zdroje.



Obrázek 2.3: Náhled UC Davis Front vyhledávače [7]

2.3.2 UC Davis front

Druhé řešení, viz obrázek 2.3, opět sestává z formuláře, tentokrát tvořeného pouze textovými poli určenými pro dotazy uživatelů. Autor opět pozoruje přítomné stránkování výsledků.

UC Davis front provozuje Katedra matematiky při univerzitě UC Davis s pravidelnými denními aktualizacemi. Například k datu 24. 4. 2015 toto řešení zahrnovalo 1032331 vědeckých článků z arXiv.org [7].

Ani UC Davis front neposkytuje řešení pro cíle stanovené v kapitole 1. Důvody jsou následující:

N1: Vyhledávač nefiltruje PDF soubory přiložené ke článkům.

N2: Vyhledávač nefiltruje RSS zdroj a negeneruje nový zdroj přizpůsobený dotazu.

2.3.3 Shrnutí

Z analýzy existujících řešení vyplývá, že ani jedno plně nepokrývá cíle práce, a tudíž že řešení problému je unikátní.

2.4 Zdroje

Následuje analýza zdrojů nezbytných pro chod výsledné aplikace.

Zřejmě bude potřeba přijímat MSC kódy formované do dotazů jako vstup. Nicméně stejně potřebné bude získávat metadata článků uvedených v aktuálním RSS zdroji a pokud bude potřeba, tak i PDF soubory ke článkům připojené. Autor by se proto nyní rád věnoval popisu těchto zdrojů, jejich umístění, struktuře a omezením spjatým s jejich získáváním.

2.4.1 Struktura zdrojů

Přístup ke všem informacím na arXiv.org je přes prohlížeč volný a v podstatě neomezený. Uživatel může ručně stáhnout prakticky jakýkoliv dokument dostupný přes libovolnou adresu `http://arxiv.org/nejaky_obsah`, na které lze

Tabulka 2.2: Ukázka struktury URL adres na arXiv.org

Tematický celek	Cesta
Obor (Matematika)	`http://arxiv.org/archive/math`
Kategorie (math.AG) – nejnovější	`http://arxiv.org/list/math.AG/new`
Kategorie (math.AG) – poslední týden	`http://arxiv.org/list/math.AG/recent`
Kategorie (math.AG) – poslední měsíc	`http://arxiv.org/list/math.AG/current`
RSS zdroj oboru (Matematika)	`http://export.arxiv.org/rss/math`
RSS zdroj kategorie (math.AG)	`http://export.arxiv.org/rss/math.AG`

něco zobrazit. Se strojovým přístupem vyvstávají jisté problémy, kterým se bude autor věnovat v podsekci 2.4.2.

Všechny URL adresy v již zmíněném tvaru jsou přehledně strukturovány, jak lze vidět v tabulce 2.2 a uživateli tak průchod a orientaci na tomto webu autoři velmi zjednodušili.

Jako alternativní zroj může sloužit zrcadlový archiv[3] strukturovaný identicky s arXiv.org.

V dalších podsekcích by se autor chtěl detailněji věnovat získávání metadat potřebných pro činnost aplikace.

2.4.1.1 Umístění článku

Nejprve by autor chtěl zanalyzovat archivaci článků, tzn. jejich umístění a možnosti jejich získávání.

Z tabulky 2.3 lze vyčíst různé způsoby, jak je možné se k jednomu z článků dostat. První dva příklady ukazují strukturu adresace článků na arXiv.org potažmo strukturu na zrcadlovém archivu zmíněném v sekci 2.4. Nicméně nejdůležitějším se jeví příklad třetí. Jedná se o arXiv API[4], jenž má za úkol umožnit vývojářům aplikací přístup k arXiv metadatům a vyhledávání [8]. Tento způsob přístupu je jednou ze dvou možností, kde druhá se nazývá OAI-PMH[5]. Přístup přes jedno z těchto API, z nichž bylo zvoleno arXiv API vzhledem k jeho účelu, se ukázal jako nutný, což bude postupně vysvětleno dále v práci.

[3] `http://lanl.arxiv.org/`
[4] `http://arxiv.org/help/api/index`
[5] `http://arxiv.org/help/oa/index`

Tabulka 2.3: Příklady umístění článků na arXiv.org

	Cesta
arXiv	`http://arxiv.org/abs/1605.04011`
lanl.arXiv	`http://lanl.arxiv.org/abs/1605.04011`
arXiv API	`http://export.arxiv.org/api/query?id_list=1605.04011`

```
1  ...
2  <entry>
3          ...
4          <title>Liouville first-passage percolation:
           ↪  subsequential scaling limit at high
5          temperature</title>
6          <author>
7          <name>Jian Ding</name>
8          </author>
9          <author>
10         <name>Alexander Dunlap</name>
11         </author>
12         ...
13         <category term="math.PR"
           ↪  scheme="http://arxiv.org/schemas/atom"/>
14         <category term="60K35, 60G60"
           ↪  scheme="http://arxiv.org/schemas/atom"/>
15         </entry>
16 ...
```

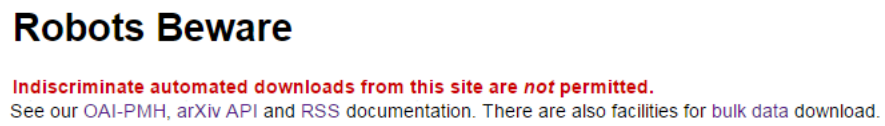
Ukázka kódu 1: Příklad výstupu aXiv API

2.4.1.2 Struktura článku

V této části by autor chtěl poukázat na význam článků v rámci aplikace a poté na hlavní nositele informace.

Na ukázce kódu 1 si lze prohlédnout, jaká metadata nabízí arXiv API zmíněné v podsekci 2.4.1.1. Nejdůležitější data pro aplikaci se nachází pod tagem *category* v atributu *term*. Zde totiž lze nalézt požadované MSC kódy.

Každý článek může mít kategorií několik. A to nejen v podobě MSC kódů, ale také interních arXiv kategorií, jako je například vidět na řádku 12 zmíněného XML kódu. Jelikož jde o stejný článek jako zmíněný v podsekci 2.4.1, interní kategorie je opět *Algebraická geometrie* neboli *math.AG*.



Obrázek 2.4: Varování arXiv pro roboty [9]

2.4.1.3 Umístění PDF dokumentů

Pro chod aplikace jsou nezbytné PDF zdroje článků. Autor do této chvíle používal pojem článek pro HTML stránky obsahující pouze metadata pravých článků. HTML stránkami autor myslí například `http://arxiv.org/abs/1605.04011`.

V této sekci by autor rád vysvětlil význam PDF souborů přiložených na těchto HTML stránkách. PDF soubory jsou samotnými publikacemi, tedy články. Za články by tedy bylo nepřesné označovat HTML stránky s metadaty, odkud lze tyto PDF soubory stáhnout. Cesta k PDF zdroji pro každý článek se velmi podobá cestě k HTML verzi, pouze řetězec */abs/* nahradí řetězec */pdf/*. Pro náš příklad tedy `http://arxiv.org/pdf/1605.04011`.

Důvod parsování i těchto PDF dokumentů je nasnadě - jelikož jsou kompletními publikacemi, měly by obsahovat hledané MSC kódy. Autoři publikací v některých případech uvedou MSC kódy pouze do PDF zdroje a nikoliv do metadat určených pro HTML stránku prezentující článek. V takových případech bude autor ze zadání uvažovat jen první stránky PDF souborů, jelikož MSC kódy bývají zpravidla uvedené na zápatí první stránky publikace. Za zmínku stojí, že PDF publikace, jejichž příslušná HTML stránka MSC klasifikaci obsahuje, tuto klasifikaci většinou neobsahují. Toto tvzení autor zakládá na otevření 10 náhodných článků[6], mezi kterými pouze 4 obsahovaly MSC kódy jak v HTML stránce, tak i v PDF publikaci.

V cílech výsledné aplikace bylo proto stanoveno získávat MSC kódy i z PDF zdrojů se záměrem pokrýt i tuto možnost a maximalizovat tak informaci, kterou ponese výstup aplikace.

2.4.2 Omezení

Při analýze autor narazil na omezení, které významně ovlivnilo návrh a implementaci. Jak lze vidět na obrázku 2.4, programátoři arXiv upozorňují před automatickým stahováním velkého množství dat jinak, než využitím zmíněných API pro to určených. Při návrhu a implementaci autor nepočítal s takovým omezením a při testování se setkal s HTTP chybovým kódem ze skupiny 400. Po několika testech byl autor schopen PDF soubory automatizovaně stáh-

[6] `http://arxiv.org/abs/1605.03975http://arxiv.org/abs/1605.04011http://arxiv.org/abs/1507.01854` ad.

Tabulka 2.4: HTTP chybové kódy: Skupina 400

HTTP kód	Popis chyby
400	Bad request.
401	Access denied.
403	Forbidden.
404	Not found. This error may occur if the file that you are trying to access has been moved or deleted.
405	HTTP verb used to access this page is not allowed (method not allowed).
406	Client browser does not accept the MIME type of the requested page.
407	Proxy authentication required.
412	Precondition failed.
413	Request entity too large.
414	Request-URL too long.
415	Unsupported media type.
416	Requested range not satisfiable.
417	Execution failed.
423	Locked error.

nout pomocí funkcí z PHP knihovny cURL[7] napsané Danielem Stenbergem. Nicméně vzhledem k upozornění výše autor toto řešení neimplementoval, jelikož nechtěl porušit žádná pravidla používání arXiv.org.

Skupina 400 indikuje chyby na straně klienta. Klient například požaduje stránku, která neexistuje nebo neposkytuje validní autentizační údaje [10].

Z tabulky kódů 2.4 autor narazil na chybu 401 při stahování HTML stránek i PDF souborů. Ve výsledné aplikaci se nakonec povedlo implementovat řešení, které splňuje cíle a vyhýbá se těmto chybám.

Z1: *HTML zdroj* – Naštěstí stahování metadat článků nebylo potřeba nijak zvlášť řešit a stačilo použít arXiv API zmiňované již v podsekci 2.4.1.1 a na obrázku 2.4. Výstup v podobě ukázky kódu 1 tohoto API je naprosto dostačující.

Z2: *PDF zdroj* – S PDF zdroji šlo o složitější řešení. Zde bohužel výsledná aplikace ztratila na funkčnosti, jelikož filtrování těchto zdrojů je podmíněné ručním stažením od uživatele aplikace. ArXiv neposkytuje žádné zdarma přístupné API byť jen pro stažení PDF metadat, nemluvě o full text verzi článku. Poskytuje pouze dvě alternativy:

[7] http://php.net/manual/en/book.curl.php

A1: Datasety přístupné pro volné stahování k dostání na stránkách Cornell University[8] ovšem nejsou vůbec relevantní vzhledem k cílům práce. Jedná se o datové zdroje z let 1992-2003.

A2: Placené Amazon S3 prostředí dostupné přes *Requester Pays Bucket* technologii. Jak na příkladu uvedeném na ukázce kódu 2 prozrazuje atribut `<yymm>` a potvrzuje help stránka arXivu[9], je tato služba aktualizována na měsíční bázi. Což nepokrývá frekvenci aktualizace RSS zdroje. Dále není možnost bezplatně otestovat Amazon S3 službu určenou pro arXiv, a proto není jisté, zda by po předplacení příslušného *Requester Pays Bucket* autor práce vůbec mohl získat informaci o MSC kategorizaci.

```
...
<file>
        <content_md5sum>1852974c8570cdafd91522ee93719ee5
        ↪  </content_md5sum>
        <filename>pdf/arXiv_pdf_0001_001.tar</filename>
        <first_item>astro-ph0001001</first_item>
        <last_item>hep-th0001208</last_item>
        <md5sum>4b5eeb603fd68bb05b9dd3341e9067fb</md5sum>
        <num_items>1751</num_items>
        <seq_num>1</seq_num>
        <size>526080000</size>
        <timestamp>2009-12-23 14:41:24</timestamp>
        <yymm>0001</yymm>
</file>
...
```

Ukázka kódu 2: Příklad PDF metadat přístupných ze služby Amazon S3 [12]

[8] http://www.cs.cornell.edu/projects/kddcup/datasets.html
[9] http://arxiv.org/help/bulk_data_s3

Kapitola 3

Návrh

3.1 UML návrh

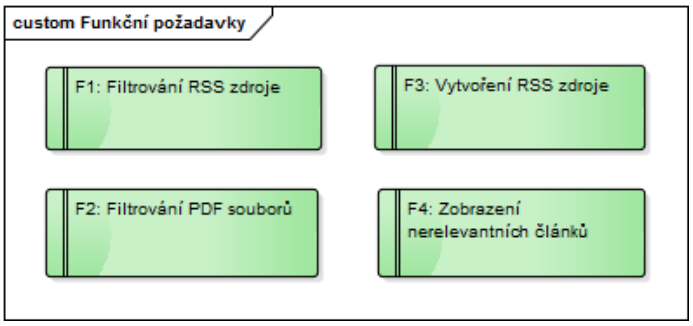
3.1.1 Model požadavků

Na úvod návrhové části by autor chtěl plynule přejít od analýzy k návrhu. Na začátku proto zmíní funkční a nefunkční požadavky na aplikaci a přejde od požadavků k modelům popisujícím vnitřní strukturu aplikace.

3.1.1.1 Funkční požadavky

Na obrázku 3.1 vyexportovaném z UML návrhu si lze prohlédnout avizované funkční požadavky. Velice věrně kopírují cíle aplikace zmíněné na začátku práce, a proto je už autor nebude více rozebírat.

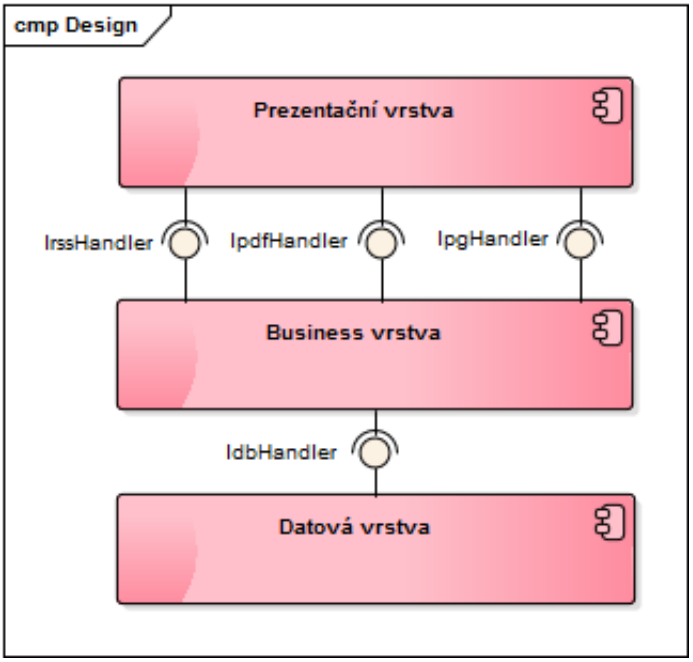
V kapitole 4 bude dále rozebráno, jak autor splnil funkční požadavky.



Obrázek 3.1: Funkční požadavky

3.1.1.2 Nefunkční požadavky

Nefunkční požadavky autor definuje tři:



Obrázek 3.2: Komponenty aplikace

N1: Vytvoření jednoduchého webového grafického rozhraní pro snadné ovládání aplikace.

N2: Optimalizace aplikace pro verze prohlížečů Google Chrome 10.0 a vyšší, Internet Explorer 10.0 a vyšší, Mozzila Firefox 4.0 a vyšší, Safari 5.1 a vyšší a Opera 10.5 a vyšší.

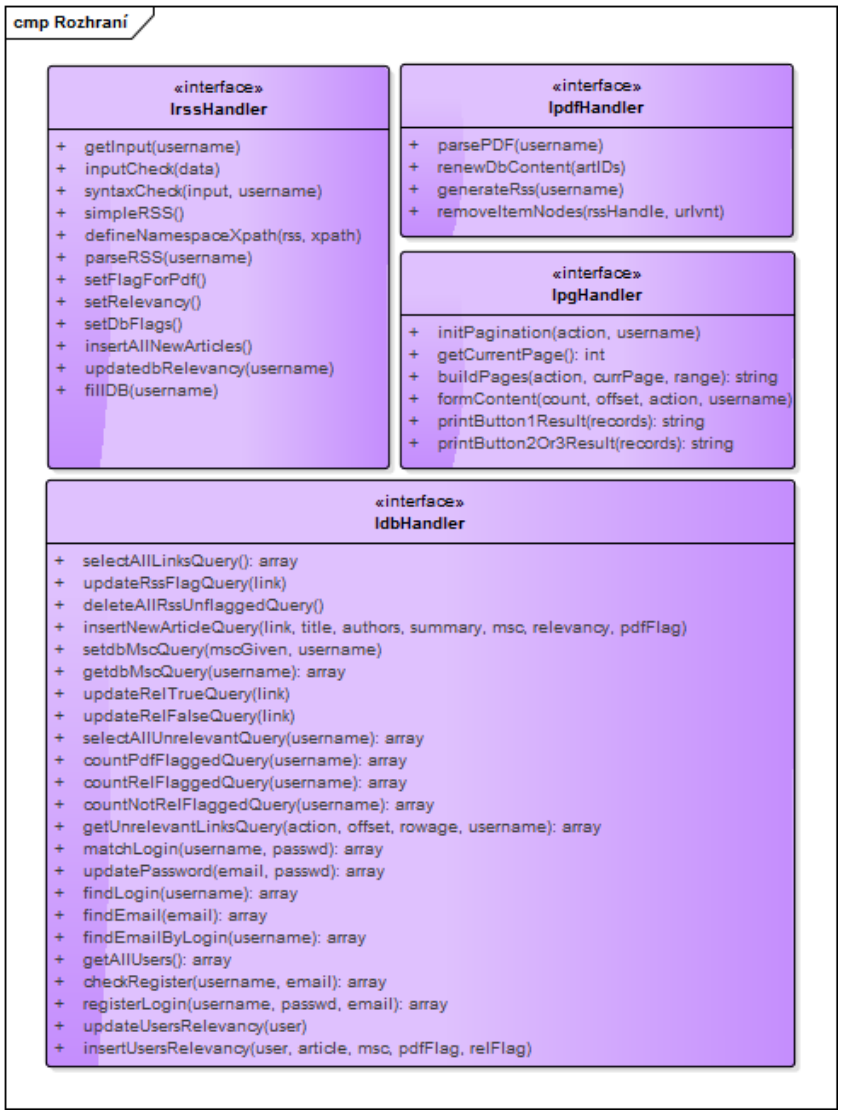
N3: Responzivní design pro možnost případného rozumného využití aplikace i na mobilních zařízeních.

I o splnění nefunkčních požadavků se bude autor blíže zmiňovat v kapitole 4.

3.1.2 Model architektury

Následující sekce je věnována popisu architektury, nad kterou bude aplikace pracovat. Při návrhu se autor snažil držet třívrstvé architektury, a tedy oddělit vrstvy prezentační, business a datovou. Autor se snaží dodržet tento model, protože většina aplikací pracujících s daty tento model dodržuje. Hlavní myšlenkou je oddělit části aplikace zabývající se prezentací dat uživateli (prezentační vrstva) a části aplikace, které se starají o její chod a jsou v pozadí (business a datová vrstva) [11].

Na výsledném návrhu na obrázku 3.2 si lze prohlédnout rozhraní zprostředkovávající komunikaci mezi vrstvami. Pomocí nich můžou abstraktní vrstvy

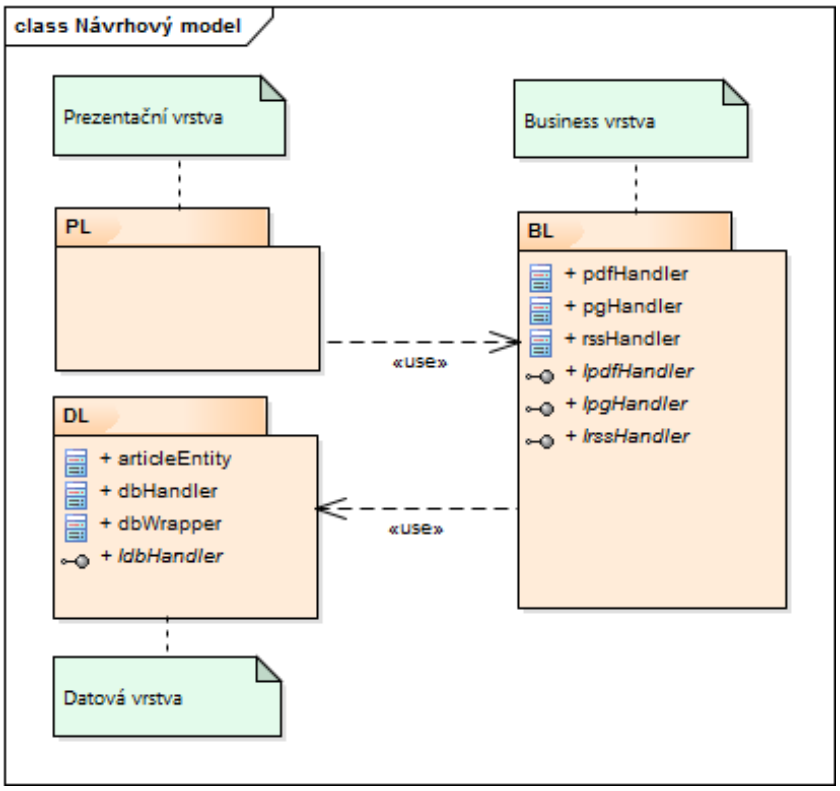


Obrázek 3.3: Ukázka rozhraní aplikace

mezi sebou konkrétně komunikovat. Rozhraní autor blíže popíše v následující sekci.

3.1.3 Rozhraní

V této sekci by autor rád dále rozvedl již avizovaná rozhraní. Pro tento účel autor vytvořil obrázek 3.3, na kterém uvádí metody využívané jednotlivými rozhraními. Z návrhu rozhraní lze vyčíst, že hlavní část aplikace se bude věnovat parsování, filtrování a ukládání informací získaných z aktuálního RSS zdroje.



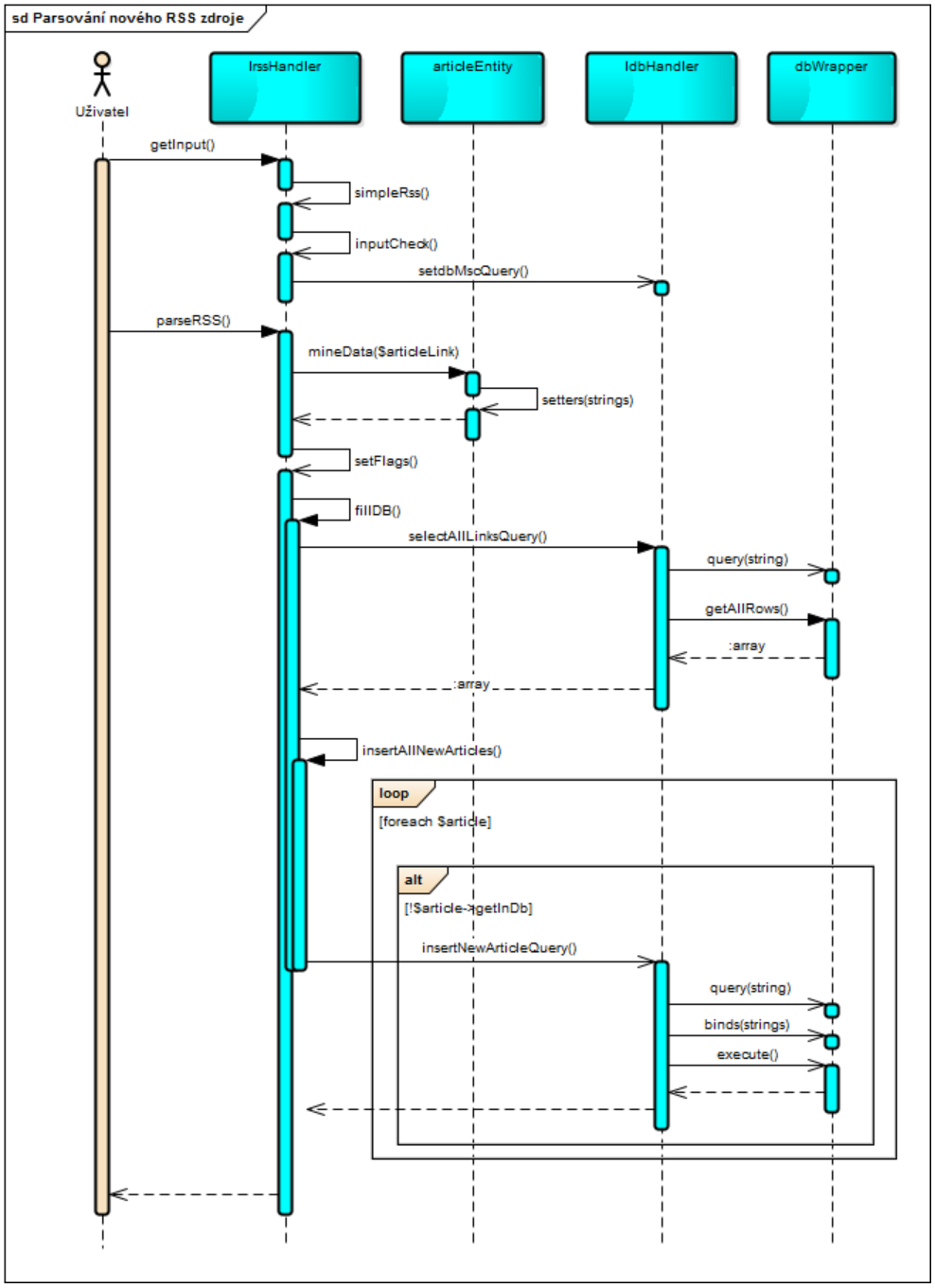
Obrázek 3.4: Návrhový model

Část odpovědná za parsování PDF dokumentů zabírá menší část aplikace a to zejména z důvodu využití knihovny PDF Parser, která implementuje veškerý parsing PDF dokumentů a jejich převádění do textové podoby. Rozhraní *IpdfHandler* se proto věnuje již jen obalení této knihovny a generování specifičtějšího RSS zdroje.

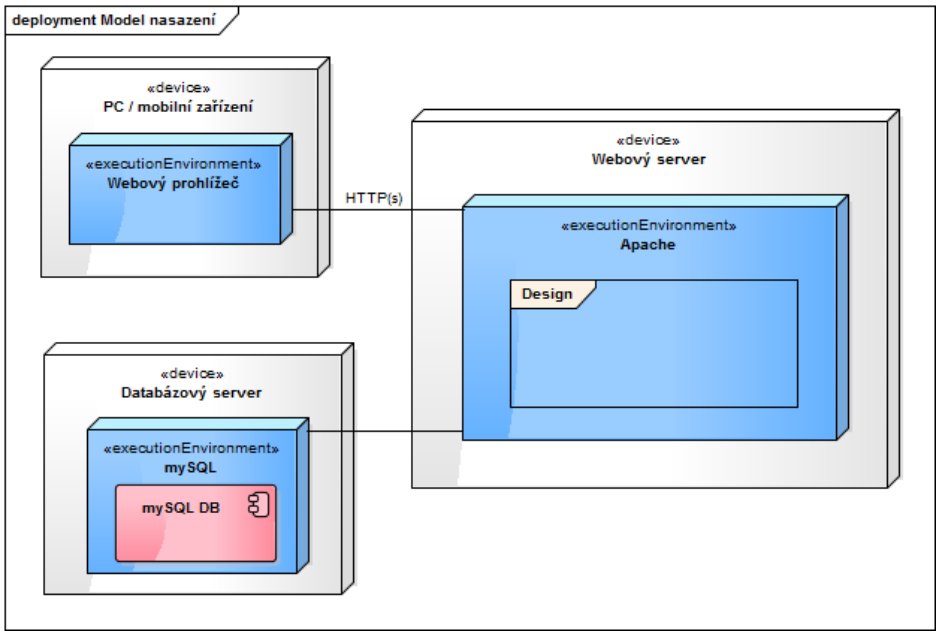
Důležité metody autor dále rozvede v kapitole 4.

3.1.4 Model komunikace

Pro přiblížení komunikace uvnitř aplikace si autor vybral tu část, která zajišťuje parsing nového RSS zdroje a uložení získaných informací pro budoucí zpracování. Na obrázku 3.5 autor připravil zkrácený přehled o třídách účastnících se tohoto procesu a hierarchii volání mezi těmito třídami potřebnými k dosáhnutí správného výsledku. Volání mezi třídami nejsou na modelu uvedena všechna a některá jsou zkrácena do jednoho. Autor použil tuto zkrácenou verzi, protože kompletní model by byl příliš velký.



Obrázek 3.5: Model komunikace



Obrázek 3.6: Model nasazení

3.1.5 Návrhový model

Návrhový model na obrázku 3.4 autor využil pro bližší zobrazení tříd spadajících do jednotlivých vrstev aplikace. Oproti již zmíněným rozhraním se v datové vrstvě objevují dvě nové třídy zodpovědné především za uložení informace o jednotlivých článcích do databáze. Zatímco třída *articleEntity* představuje strukturu uložení metadat článku, třída *dbWrapper* vyváří obalení PHP databázových volání pro rozhraní *IdbHandler*.

3.1.6 Model nasazení

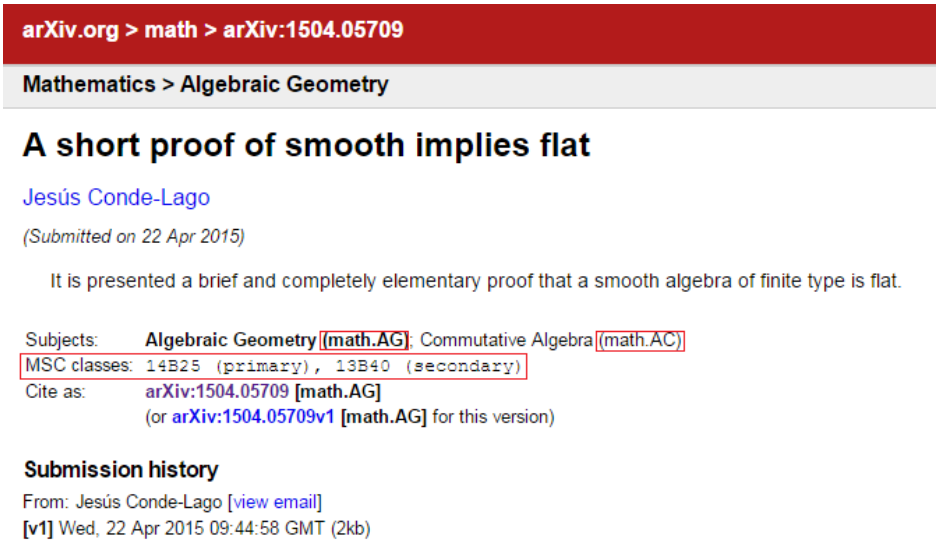
V poslední části věnující se UML návrhu chce autor znázornit nasazení aplikace. MySQL databáze pro aplikaci se nachází na stejném serveru.

Příloha s výslednou aplikací obsahuje instalační příručku, po jejímž splnění bude možné aplikaci spouštět lokálně.

3.1.7 Požadavky na uživatele

Doposud byl věnován prostor zejména analýze a návrhu ze strany aplikace. Ke konci této kapitoly chce autor popsat požadavky kladené na uživatele pro smysluplné využití aplikace:

P1: Znalost MSC a jiných kategorií používaných v metadatech článků z arXiv.org.


arXiv.org > math > arXiv:1504.05709

Mathematics > Algebraic Geometry

A short proof of smooth implies flat

Jesús Conde-Lago

(Submitted on 22 Apr 2015)

It is presented a brief and completely elementary proof that a smooth algebra of finite type is flat.

Subjects: Algebraic Geometry (math.AG); Commutative Algebra (math.AC)
MSC classes: 14B25 (primary), 13B40 (secondary)
Cite as: arXiv:1504.05709 [math.AG]
(or arXiv:1504.05709v1 [math.AG] for this version)

Submission history

From: Jesús Conde-Lago [view email]
[v1] Wed, 22 Apr 2015 09:44:58 GMT (2kb)

Obrázek 3.7: Náhled článku z arXiv.org [13]

P2: Znalost toho, jaké výstupy od aplikace očekávat - výsledek je určen především zasvěceným uživatelům, proto si tento požadavek autor může dovolit.

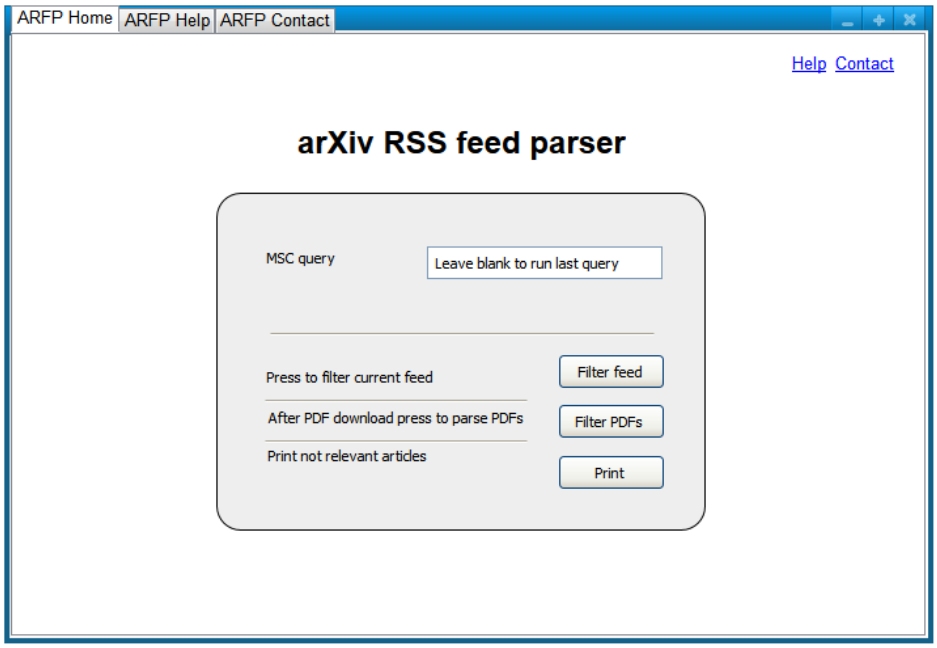
P3: Ruční dodatečné stažení PDF souborů pro další kontrolu relevance.

Na obrázku 3.7 lze vidět vyznačené MSC a další kategorizace. Tyto kategorizace je uživatel povinen znát zcela a nebo minimálně tu jejich část, kterou chce filtrovat. Vzhledem k tomu, že jako vstup aplikace se očekává např. jeden MSC kód nebo jejich množina, jde opravdu o nezbytnou znalost. Uživatelé bez této znalosti nemohou práci plně využít. Odkaz na celý MSC standard byl již zmíněn v sekci 2.1.

Správné použití aplikace je také podmíněno znalostí toho, co lze očekávat jako její výstup. V tomto případě jde o RSS zdroj zpracovaný do té podoby, kde už se nevyskytují články bez požadované klasifikace. I tato znalost je zásadní, a to hlavně pro využitelnost aplikace daným uživatelem. Bez ní může uživatel aplikaci sice používat smysluplně (nutnou podmínkou je požadavek P1), ale bez znalosti formy výstupu nebude takovému uživateli aplikace generovat žádný smysluplný obsah.

3.2 Prototyp

Na konec této kapitoly si autor přichystal obrázek 3.8 s prototypem výsledné aplikace. Její uživatelské rozhraní nabízí velmi jednoduché ovládání všech dostupných funkcí. Jakmile bude aplikace nahrána na server, vytvoří se soubor



Obrázek 3.8: Prototyp Home page aplikace

pro plánované spouštění v 4:00 středoevropského času. Díky tomu akce spuštěná zmáčknutím prvního tlačítka nebude muset stahovat všechny soubory pro sběr metadat, která už jsou pro daný den uložená v databázi. Na verzi aplikace přiložené k práci bude nadále nastavena akce pro stáhnutí všech metadat, jelikož při lokálním použití aplikace si musí každý uživatel případné plánované spouštění povolit sám.

Kapitola 4

Implementace

V této kapitole se bude autor snažit přiblížit programovací specifika aplikace, plnění funkčních a nefunkčních požadavků těmito specifiky a dokumentaci kódu použitého pro sestavení aplikace. Autor provede diskusi použitých technologií následovanou praktickými ukázkami kódu.

4.1 Použité technologie

4.1.1 XML

O XML struktuře metadat v arXiv API už byla krátká řeč v kapitole 2. Zde by autor chtěl stručně rozebrat XML jazyk jako takový a motivaci pro jeho použití, případně podmínky spjaté s jeho použitím.

V jisté míře autor neměl na vybranou, jelikož musí zpracovávat RSS zdroj, který je sestavován v konvencích tohoto jazyka. Nicméně jazyk XML, vyvinutý ze standardu SGML byl vytvořen především pro lehce srozumitelné zpracování velkých elektronických publikačních systémů [14]. V současnosti význam jazyka XML stále roste, a to především v rámci výměny dat na webu i jinde. Obě dostupné verze XML 1.0 i XML 1.1 byly vyvíjeny s důrazem na snadnou implementaci a komunikaci s obecným SGML standardem a také HTML jazykem [15, 16]. Díky těmto předpokladům, se kterými byly a jsou verze XML jazyka vyvíjeny, autor pouze těžil z výhod a nebyl omezován požadavky.

Zmíněný RSS zdroj ovšem, oproti výstupům arXiv API, viz ukázku kódu 1 v kapitole 2, není jen XML dokument. Syntaxe dokumentu je ještě specifikována do RSS podoby.

4.1.1.1 RSS 1.0

RSS je specifická, zjednodušená forma popisu metadat. Jde o aplikaci XML jazyka respektující RDF konvence. RDF představuje podobně jako XML obecný

jazyk pro popis elektronických dokumentů na internetu. Koneckonců XML jazyk je použit pro generování RDF grafů, přesněji XML/RDF jazyk [17].

Výsledkem označování dokumentů pomocí RDF syntaxe může být jejich spojování do Sémantického webu, což se dá popsat jako rozšíření současného webu, v němž informace mají přidělen dobře definovaný význam lépe umožňující počítačům a lidem spolupracovat. Jde tedy o to, aby data prezentovaná na internetu měla přesně definovaný význam a dovolovala do značné míry automatizované (strojové) zpracování [19].

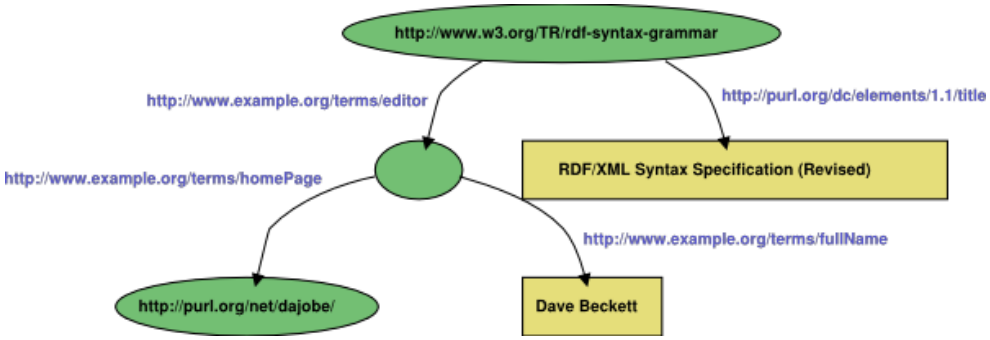
Na ukázce 3 je jazykem XML/RDF popsaná jedna cesta grafem (po zelených uzlech) z obrázku 4.1. Z kódu lze vyčíst označení `rdf:Description` pro uzel, a to jak prázdný, tak i ten s URI identifikátorem, a `ex:literál` pro hrany nebo uzly obsahující literál v podobě řetězce, kde *ex* určuje jmenný prostor (modul).

Podle určitých pravidel k nahlédnutí na stránkách organizace W3C[10] lze pomocí zkracování a využití atributů potažmo hodnot atributů získat minimalizovanou, *kompletní* formu, která je uvedená na ukázce kódu 4. Daný kód popisuje všechny možné cesty v příslušném grafu.

```
1  <rdf:Description>
2    <ex:editor>
3      <rdf:Description>
4        <ex:homePage>
5          <rdf:Description>
6          </rdf:Description>
7        </ex:homePage>
8      </rdf:Description>
9    </ex:editor>
10 </rdf:Description>
```

Ukázka kódu 3: Příklad základního, nekompletního XML/RDF popisu RDF grafu

[10] `http://www.w3.org/TR/REC-rdf-syntax/#section-Syntax`



Obrázek 4.1: Příklad RDF grafu [18]

```
<?xml version="1.0"?>
<rdf:RDF
 ↪ xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:dc="http://purl.org/dc/elements/1.1/"
                xmlns:ex="http://example.org/stuff/1.0/">
  <rdf:Description
   ↪ rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
      dc:title="RDF/XML Syntax Specification (Revised)">
    <ex:editor>
      <rdf:Description ex:fullName="Dave Beckett">
            <ex:homePage
             ↪ rdf:resource="http://purl.org/net/dajobe/" />
      </rdf:Description>
    </ex:editor>
  </rdf:Description>
</rdf:RDF>
```

Ukázka kódu 4: Příklad kompletního XML/RDF popisu RDF grafu

4.1.1.2 arXiv RSS

ArXiv pro své RSS zdroje používá verzi RSS 1.0, která má jistá specifika (citováno z [20]):

S1: Celý zdroj obaluje element `<rdf:RDF> ... </rdf:RDF>`.

S2: Každá položka v RSS zdroji, pro nás článek, označená elementem `<item>` obsahuje v atributu *rdf:about* identifikátor zpravidla shodný s informací v elementu `<link>`, který určuje cestu k článku.

S3: Každý zdroj obsahuje položku `<items>`, která obsahuje seznam všech položek přítomných ve zdroji.

S4: Některá metadata využívají atribut *rdf:resource* pro uchování cest namísto ukládání těchto cest v dalších elementech.

Konkrétně pak jde o RSS 1.0 mimo jiné s modulem *Dublin Core Module*, což je množina metadat vyvinutá knihovníky a informatiky pro standardizaci častých metadat užitečných mimo jiné pro popis dokumentů. Jak pro celý zdroj, tak i pro jednotlivé položky zdroje [20].

Nyní by autor rád zakomponoval do práce část RSS zdroje[11] ze dne 2. 5. 2015, k vidění na ukázce kódu 5. Jde o zdroj, se kterým aplikace přímo pracuje. Na ukázce jsou patrná zmiňovaná specifika RSS 1.0, *Dublin Core Module* a také odkaz na RDF jmenný prostor *xlmns:rdf*.

4.1.1.3 Atom

Jako standardizovaná alternativa k RSS 1.0 a RSS 2.0 byl vyvinut Atom. Vzhledem k tomu, že ani jeden ze zmíněných RSS formátů není standardem, vyvinula společnost *IETF Working Group* formát také založený na XML s podobnou funkcí jako RSS [20].

Příklad Atom formátu již autor uvedl při popisování arXiv API v sekci Analýza na ukázce kódu 1. V podsekci 4.1.1 autor označil tento výstup obecně jako XML formát, což bylo v této podsekci upřesněno.

4.1.1.4 Shrnutí

Jak bylo vysvětleno, formáty XML, RSS 1.0 a Atom práce využívá především z pohledu čtení výstupů poskytovaných od archivu arXiv. Aplikace obsahuje použití těchto formátů ve formě zápisu pouze při vytváření konkrétnějšího RSS 1.0 zdroje po vyfiltrování relevantních článků vzhledem k MSC dotazu.

4.1.2 HTML

Jazyk HTML autor využil především pro vytvoření struktury webových stránek aplikace. Jde o stěžejní jazyk pro tvorbu dokumentů a aplikací připravených pro kohokoliv k použití především prostřednictvím internetu [21].

Obsah HTML dokumentu, podobně jako u XML formátu, označují specifické značky jako například `<title>...</title>`, `<div>...</div>` apod.

[11] `http://export.arxiv.org/rss/math`

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://purl.org/rss/1.0/"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:syn="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:admin="http://webns.net/mvcb/">
  <channel rdf:about="http://arxiv.org/">
    <title>math updates on arXiv.org</title>
    <link>http://arxiv.org/</link>
    ...
    <dc:language>en-us</dc:language>
    <dc:date>2015-04-30T20:30:00-05:00</dc:date>
    ...
    <items>...</items>
    <image rdf:resource="http://arxiv.org/icons/sfx.gif"/>
  </channel>
    ...
  <item rdf:about="http://arxiv.org/abs/1504.07982">
    <title>
      The equality of capacity and modulus of condenser in
      Sub-Finsler space. (arXiv:1504.07982v1 [math.CA])
    </title>
    <link>http://arxiv.org/abs/1504.07982</link>
    ...
    <dc:creator>
      <a href= "http://arxiv.org/find/math/1/au:+Dymchenko_
      Y/0/1/0/all/0/1">Yu. V. Dymchenko</a>
    </dc:creator>
  </item>
  ...
</rdf:RDF>
```

Ukázka kódu 5: Část arXiv RSS zdroje

4.1.3 CSS

Mechanismus pro přidávání stylů webovým dokumentům se nazývá CSS [22]. Hlavní myšlenkou CSS je odlišit prezentaci a strukturu dat. HTML dokument by měl poskytovat strukturu, zatímco CSS určí způsob prezentace [23]. Autor využil CSS především pro určení způsobu prezentace webových stránek aplikace.

4.1.4 PHP

Pod zkratkou PHP se skrývá velice populární skriptovací jazyk speciálně uzpůsobený pro tvorbu webových aplikací, který může být vkládán přímo do HTML dokumentů pomocí uzavření mezi instrukce `<?php` a `?>` [24].

Naprostá většina práce je napsána v jazyce PHP, přičemž část kódu využívá možnosti vkládat PHP skript dovnitř HTML dokumentů.

4.1.5 MySQL

MySQL patří mezi tzv. relační databáze, tedy mezi databáze založené na tabulkách, kde každá tabulka obsahuje položky se stejnými vlastnostmi neboli atributy. Pod pojmem relační si lze představit vztah (anglicky relation), a to jak mezi tabulkami, tak i mezi entitami v jedné tabulce.

Pojem databáze není úplně přesný, přesnějším označením by bylo RDBMS, což v překladu znamená administrační systém pro relační databázi. RDBMS mimo jiné umožňuje spojovat několik dotazů do transakcí, kdy se série dotazů vykoná vždy celá nebo vůbec [25]. Vlastnosti databázového stroje ohledně transakcí se často vyjadřují zkratkou ACID (citováno z [25]):

A: *Atomicicty* – Operace v transakci se provedou jako jedna atomická (nedělitelná) operace.

C: *Consistency* – Stav databáze po dokončení transakce je vždy konzistentní, tedy validní podle všech definovaných pravidel a omezení.

I: *Isolation* – Operace jsou izolované a navzájem se neovlivňují.

D: *Durability* – Všechna data se okamžitě zapisují na trvalé úložiště, například pevný disk. V případě přerušení provozu RDBMS se databáze vrátí do stavu těsně před výpadkem.

Autor v aplikaci využívá MySQL především pro uložení metadat článků získaných z RSS zdroje, ale také i pro uložení posledního MSC dotazu. Aplikace při prázném dotazu sáhne do databáze pro poslední dotaz.

Autor zvažoval, zda by nestačilo využít uložení dat ve formě např. XML souboru, došel ovšem k názoru, že použití MySQL databáze je zejména díky ACID vlastnostem transakcí vhodnější. Také na přehlednosti kódu pro případná budoucí rozšíření spíše přidají srozumitelné SQL příkazy, než načítání dat ze souborů.

Konkrétně pak autor upřednostnil MySQL databázi před jinými možnostmi zejména kvůli její rozšířenosti a již nabyté znalosti MySQL technologie.

4.1.5.1 phpMyAdmin

Pro vytvoření databáze a tabulek, prohlížení jejich obsahu a vygenerování SQL *CREATE* skriptů autor použil webové rozhraní phpMyAdmin. Jde o nástroj pro administraci celého MySQL serveru nebo třeba jen určité databáze. Pro správné fungování je potřeba zejména správně nastavit MySQL uživatele, který bude moci prohlížet/zapisovat jen do daných databází [26].

4.1.6 Composer

Nástroj k administraci závislostí v PHP projektu, Composer, pomocí souboru *composer.json* s určitou strukturou jednoduše nainstaluje knihovny potřebné pro projekt a nastaví závislosti [27]. Autor použil tento nástroj především pro instalaci knihovny *PDF Parser v0.9.23*. Composer tedy přímo v implementaci aplikace nefiguruje, ale byl součástí nastavení závislostí potřebných pro použití knihovny nezbytné pro parsování PDF dokumentů s články.

4.1.7 PDF Parser

Autor použil knihovnu PDF Parser[12] pro získávání textu z PDF dokumentů. Jde o knihovnu napsanou v jazyce PHP od Sébastiena Malota.

PDF Parser autor zakomponoval do projektu pomocí nástroje Composer viz minulá podsekce, a to konkrétně vytvořením souboru *composer.json* v podobě viz ukázku kódu 6. Následující příkaz z ukázky kódu 7 spuštěný z příkazové řádky nad adresářem se souborem *composer.json* nastaví v projektu příslušné závislosti.

```
{
  "require": {
    "smalot/pdfparser": "*"
  }
}
```

Ukázka kódu 6: Composer.json soubor pro instalaci knihovny PDF Parser

```
$ composer update smalot/pdfparser
```

Ukázka kódu 7: Příkaz pro stažení knihovny

4.1.8 Regulární výrazy

Za regulární výraz se považuje řetězec se specifickou syntaxí udávající vzor převážně pro porovnávání s jinými řetězci. Výsledkem porovnání může být

[12] `http://www.pdfparser.org/documentation`

Tabulka 4.1: Použité speciální sekvence regulárních výrazů PHP

Speciální znaky	Význam
\d	Libovolná číslice
/	Úvodní i uzavírací znak výrazu
\d{2}	Dvě libovolné číslice za sebou
[A-Z]	Libovolné ASCII velké písmeno

shoda nebo neshoda regulárního výrazu s testovaným řetězcem, přičemž shoda může nastat pro více než jeden testovaný řetězec. Syntaxi regulárního výrazu tvoří obyčejné znaky, jako například číslice nebo písmena, a speciální znaky s vlastním významem [28].

Autor využil regulární výrazy pro indikování, zda článek bude potřebovat další zpracování v rámci určení relevance nebo ne. Regulární výrazy jsou tedy využity pro nastavování příznaku, zda je dále potřeba parsovat i PDF dokument článku, nebo nikoliv. Pokud v metadatech článku nebyla uvedena žádná MSC klasifikace, je potřeba ho dále zpracovat.

Aplikace využívá regulární výrazy kompatabilní s jazykem Perl. Jde o jednu ze dvou možností práce s regulárními výrazy v PHP, přičemž druhá možnost (POSIX standard) byla od verze PHP 5.3.0 označena jako *deprecated* neboli zastaralá [29]. Regulární výraz použitý v aplikaci[13] využívá hned několik speciálních sekvencí znaků definovaných pro regulární výrazy Perlu, jejich vysvětlení lze nalézt v tabulce 4.1.

4.1.9 Přihlášení

Aby mohlo aplikaci používat více uživatelů, zvolil autor cestu registrování a přihlašování. Pro správný běh aplikace byly zprovozněny funkce:

R: *Registrace* - Uživatelé se mohou sami zaregistrovat. Hesla jsou uložena v databázi jako hash kód se solí. V tomto směru byla využita PHP funkce `password_hash()`.

P: *Přihlášení* - Uživatelé se samozřejmě po registraci mohou přihlásit.

Z1: *Zapomenuté heslo* - Aplikace poskytuje možnost si obnovit heslo, pokud ho uživatel zapomněl.

Z2: *Změna hesla* - Jelikož akce zapomenuté heslo vyústí ve vygenerování náhodného nového hesla, může si ho (nejen tehdy) uživatel kdykoliv změnit.

[13] /\d{2}[A-Z]\d{2}/

O: *Oznámení* - Za pomoci volně dostupné knihovny PHPmailer[14] při splnění licenčních podmínek[15] aplikace rozesílá nezbytné informace ohledně předchozích bodů na email zadaný při registraci.

4.2 Zdrojové kódy a skripty

Poslední sekce práce se bude věnovat konkrétním implementačním metodám využitým v aplikaci. Na příkladech autor odůvodní nezbytnost některých specifik aplikace, případně poukáže na zajímavosti ve zdrojovém kódu.

Jak bylo již zmíněno, aplikace je napsána převážně v jazyce PHP. Většina ukázek proto také bude z tohoto jazyka. Na úvod sekce by autor chtěl zmínit funkční a nefunkční požadavky a jejich splnění.

4.2.1 Požadavky – shrnutí

4.2.1.1 Funkční požadavky

F1: *Parsování RSS zdroje* – Rozhraní *IrssHandler* poskytuje kompletní sadu metod pro získání všech informací, které aplikace potřebuje pro uložení do databáze, případně další zpracování. Metody tohoto rozhraní se starají například o stáhnutí RSS zdroje a jeho převedení standardní PHP funkcí do SimpleXML objektu, který nabízí jednoduché přistupování k prvkům článků (v této fázi především odkaz na článek obsahující jeho jednoznačné ID) pomocí šipkové konvence. Požadavek byl plně pokryt autorovou vlastní implementací *IrssHandler* rozhraní.

F2: *Filtrování PDF souborů* – Parsování PDF zdrojů článků zajišťuje rozhraní *IpdfHandler*, které k tomu využívá zmíněnou knihovnu *PDF Parser*. Zdrojové PDF dokumenty článků nejsou volně ke stažení se zdůvodněním od arXiv založeném na faktu, že arXiv nemůže poskytnout licenci ke všem článkům, které schraňuje. Bohužel pro PDF soubory administrátoři archivu neposkytují volné API podobné jako arXiv API pro volné stahování metadat článků. Nelze tedy implementovat full textové vyhledávání v souladu s podmínkami užití arXiv.org jinak, než že autor práce nechá na uživateli ruční stažení PDF dokumentů pro parsing a uloží je do dané složky, ve které je aplikace očekává. PDF dokumenty uložené v této složce po vyvolání akce k jejich parsování budou převedeny do textu (alespoň jejich první strana, na které mohou být uvedené MSC kódy) a tento text bude porovnán s regulárním výrazem reprezentujícím MSC kód. V případě nálezu shody nastává k aktualizaci příznaku relevance daného článku v databázi. Finálně je vytvořen nový RSS zdroj

[14] `https://github.com/PHPMailer/PHPMailer`
[15] `http://www.gnu.org/licenses/lgpl-2.1.html#SEC4`

obsahující pouze relevantní články vzhledem k vstupnímu dotazu, čímž rozhraní *IpdfHandler* pokrývá i třetí funkční požadavek.

Při automatickém spouštění na serveru se nový RSS zdroj vytvoří bez potřeby vyvolávat akci k parsování PDF dokumentů.

F3: *Vytvoření nového RSS zdroje* – Viz popis řešení požadavku F2.

F4: *Zobrazení nerelevantních článků* – Poslední požadavek již představuje jen malý přídavek k řešení předchozích požadavků. Po splnění prvního požadavku je aplikace ve stavu, kdy je databáze doplněna novými články od poslední aktualizace (při automatickém spouštění tedy od posledního pracovního dne). Podle MSC dotazu uživatele, který je uchován v databázi a aplikuje se, dokud uživatel dotaz nezmění, jsou nastaveny příznaky. Pomocí příznaků, které ještě může aktualizovat parsování PDF dokumentů v případě, že shoda dotazu a klasifikace byla nalezena až zde, aplikace v další akci pozná relevantní a nerelevantní obsah. Pro splnění tohoto požadavku proto aplikace pouze vykoná poslání nerelevatních článků vzhledem k dotazu na výstup.

Funkční požadavky byly splněny.

4.2.1.2 Nefunkční požadavky

N1: Webové grafické rozhraní aplikace poskytuje velmi jednoduché ovládání pomocí několika tlačítek, přičemž jedno z nich pouze vykoná návrat na úvodní stránku a vymaže z URL adresy informace použité pro stránkování výsledků ostatních akcí. Stránkování samotné využívá konvenční značení, kdy pro posun dopředu/zpět příp. na začátek/na konec používá znaky < a >.

Pro uživatele autor dále připravil stránku *Help*, kde může uživatel najít odpovědi na předpokládané otázky k ovládání aplikace.

N2: Při vytváření CSS stylů pro správné zobrazení stránek aplikace autor elektronicky konzultoval internetové stránky w3schools[16], na nichž lze dohledat, které verze prohlížečů daný CSS příkaz podporují. Díky těmto konzultacím autor dodržel nefunkční požadavek N2. Hraničními příkazy v tomto jazyce se stala kombinace příkazů: příkazy pro stínování prvku/textu uvnitř prvku *box-shadow: none;* a *text-shadow: none;*. Ostatní příkazy byly podporovány od dřívějších nebo stejných verzí jako tyto.

N3: Pro splnění posledního nefunkčního požadavku autor využil možnost CSS příkazu *@media*. Na ukázce kódu 8 autor uvádí část kódu zodpověd-

[16] `http://www.w3schools.com/cssref/`

nou např. za zvýšení relativní šířky formuláře elementu s identifikátorem *FormBox* od maximálního rozlišení 700px a menšího na 70 %.

```
@media screen and (max-width: 700px){
        #FormBox{
                width: 70%;
        }

        ...

        #footer p:not(:last-child) {
                padding-right: 15%;
        }
}
```

Ukázka kódu 8: CSS příkaz pro responzivní design

Nefunkční požadavky byly stejně jako funkční splněny.

4.2.2 Bezpečnost

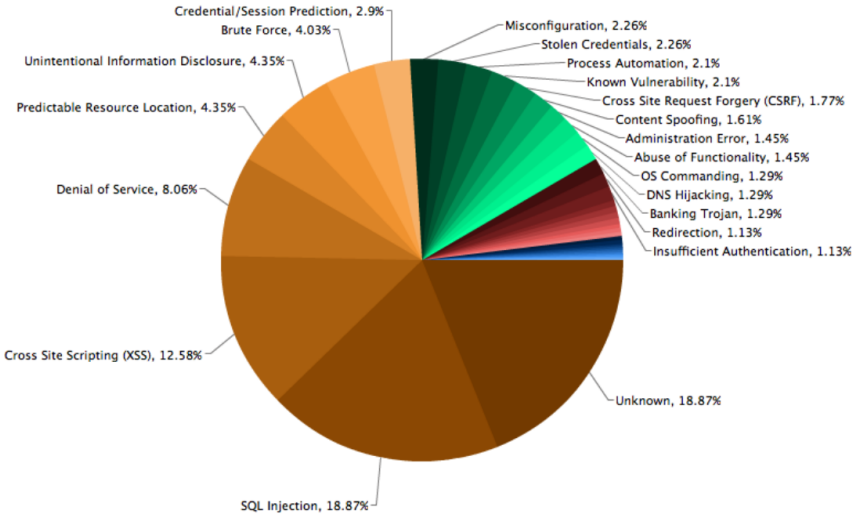
Autor sice nepředpokládá vysokou návštěvnost webových stránek aplikace, a to především od neznalých uživatelů, přesto se rozhodl bránit proti určitým typům útoků na data v databázi a nebo škodlivému skriptování. Podle grafu na obrázku 4.2 převzatého z webu acunetix[17] jde o dva nejčastější typy útoků na webové stránky. Zároveň není příliš složité se proti oběma útokům bránit.

4.2.2.1 XSS

Při útoku typu XSS dochází k vložení škodlivé informace prostřednictvím URL nebo formuláře do kódu stránky. K úspěšnému útoku dojde tehdy, pokud tvůrce napadené webové stránky nijak nevaliduje vstup přijatý například z formuláře na své stránce a například JavaScript příkazem `print` takový vstup ihned zobrazí. Webová stránka je tehdy úspěšně napadena a škodlivý kód ze vstupu se provede. Může jít jak o banální kód, který pouze zobrazí tučný text, například `<b>pokus</b>` , tak třeba i o spuštění externího JavaScriptového škodlivého souboru[31].

Autor možnosti napadení detekoval na několika místech v aplikaci a všude vstup validuje. Například po přijetí MSC dotazu od uživatele, tento dotaz projde metodou *inputCheck* viditelnou na ukázce kódu 9, která vstupní text ořízne o mezery na začátku a konci řetězce, následně metoda smaže všechna zpětná lomítka, která nejsou v řetězci zdvojená. Nakonec metoda nahradí

[17] `http://www.acunetix.com/websitesecurity/cross-site-scripting/`



Obrázek 4.2: Graf nejčastějších webových útoků [30]

všechny HTML významné znaky za jiné, nevýznamné znaky. Výsledkem metody je řetězec, který nevloží škodlivou HTML značku do stránek aplikace a nevykoná tak škodlivý kód.

V aplikaci autor dále implementoval stránkování výsledků, které si předává parametry přes URL pomocí metody $_GET. Předchozí příklad při předávání MSC dotazu používal metodu $_POST, která posílá informace z formuláře k dalšímu zpracování skrytě a je vhodná pro interní informace aplikace. Metoda $_GET naproti tomu předává informace přímo v URL adrese [32]. Po potvrzení formuláře využívající tuto možnost se vyplněná data zobrazí v URL adrese. Autor tento způsob využil při stránkování, protože při přecházení mezi stránkami docílil jednoduchého předání informace v podobě odkazu na tu stránku, na kterou uživatel klepnul. Na ukázce kódu 10 autor uvádí k nahlédnutí URL z lokálního testování, kde lze vidět proměnnou *action* s hodnotou *pdfParse* a proměnnou *page* s hodnotou *1*, což dohromady určuje, jakým způsobem má aplikace daný obsah stránkovat a na jaké stránce z celého obsahu se uživatel momentálně nachází.

Obě hodnoty jsou v aplikaci kontrolovány. Hodnota *page* na to, zda se jedná o číslo, a hodnota *action* na to, zda jde o jednu ze tří aplikaci známých hodnot. Při nerozpoznání správných hodnot aplikace buď ohlásí chybné URL (pro chybu v proměnné *action*), a nebo zobrazí pouze obsah stránky 1 a vnitřně si zapamatuje, že hodnota proměnné *page* je 1.

```
function inputCheck($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```

Ukázka kódu 9: Ochrana proti XSS útoku

```
http://localhost/.../index.php?action=pdfParse&page=1
```

Ukázka kódu 10: Předání $_GET proměnných

4.2.2.2 SQL Injection

SQL Injection znamená útok na databázi prostřednictvím vstupu aplikace, například opět vstupu textového. Jde o vstup v takové podobě, která předpokládá zpracování v SQL (MySQL) jazyce. Chytrým vstupem za znalosti tohoto jazyka pak útočník může zadat informace například pro zobrazení všech záznamů z databáze, kdy použije vždy pravdivé tvrzení pro SQL (MySQL) klauzuli WHERE [33].

Obranou před takovým typem útoku můžou být předpřipravené dotazy. Myšlenkou této technologie je připravit dotaz pro databázi s dočasnými proměnnými, do kterých budou vloženy pravé, již zkontrolované, proměnné. Každý parametr se do připraveného dotazu přiloží metodou *bindParam*, alespoň co se týká třídy PDO, která implementuje rozhraní mezi PHP aplikací a databázovým serverem [34]. Autor využívá třídu PDO pro komunikaci s webovým serverem, ovšem nekontroluje vstup od uživatele proti databázi. Jediná data, která figurují v MySQL dotazech použitých autorem pro běh aplikace, jsou důvěryhodná data ze serverů arXiv. Autor přesto, pro opatrnost, předpřipravené dotazy využívá a pro ilustraci uvádí metodu na vložení článku do databáze na ukázce kódu 11 z rozhraní *IrssHandler*.

```
public function insertNewArticleQuery($link, $title, $authors,
↪  $summary, $msc, $relevancy, $pdfFlag) {
        $this->handler->query('INSERT INTO article (
        path, title, authors, summary,
        classes, relFlag, pdfFlag, rssFlag
        ) VALUES (
        :link, :title, :authors, :summary,
        :msc, :relFlag, :pdfFlag, 1
        )');

        $this->handler->bind(":link", $link);
        $this->handler->bind(":title", $title);
        $this->handler->bind(":authors", $authors);
        $this->handler->bind(":summary", $summary);
        $this->handler->bind(":msc", $msc);
        $this->handler->bind(":relFlag", $relevancy);
        $this->handler->bind(":pdfFlag", $pdfFlag);

        $this->handler->execute();
}
```

Ukázka kódu 11: Ochrana proti SQL Injection

4.2.3 Vytváření vlastních RSS zdrojů

Na konec práce si autor připravil ukázky kódu 12 a 13 z rozhraní *IpdfHandler*, ve které je uveden způsob, jakým autor vytváří z aktuálního arXiv matematického RSS zdroje vlastní, MSC dotazem omezený zdroj.

Na ukázce si lze všimnout práce s RSS jmennými prostory, které je potřeba registrovat prakticky v každé úrovni zanoření, atributy uzlů nevyjímaje. Dále si lze všimnou šipkové konvence SimpleXML objektů, která umožňuje snadný přístup k uzlům a atributům dokumentu.

Funkce na ukázkách, `function removeItemNodes(rssHandle,urlvnt)`, s parametry zdrojového RSS souboru a asociovaného pole obsahujícího nerelevantní články pro právě přihlášeného uživatele, vytvoří výsledný profiltrovaný RSS soubor.

```
function removeItemNodes($rssHandle, $urlvnt) {
  //article count minus channel and items children (-2)
  $count = $rssHandle->count() - 2;

  for( ;$count > 0; --$count) {
    foreach ($urlvnt as $arr) {
      if (!strcmp((string)$rssHandle->item[$count-1]->link,
      ↪  $arr['article'])) {
        unset($rssHandle->item[$count-1]->{0});
        break;
      }
    }
  }
  ...
}
```

Ukázka kódu 12: Vytváření vlastního RSS zdroje1

```
...
//Multiple namespace registration while going through the PDF
↪  graph
$Seq = $rssHandle->channel->items->children
↪  ('http://www.w3.org/1999/02/22-rdf-syntax-ns#');
$lis = $Seq->children
↪  ('http://www.w3.org/1999/02/22-rdf-syntax-ns#');
$cnt = $lis->count();

while($cnt) {
  $attr = $lis[--$cnt]->attributes
  ↪  ('http://www.w3.org/1999/02/22-rdf-syntax-ns#')->resource;

  foreach ($urlvnt as $arr) {
    if (!strcmp((string)$attr[0], $arr['article'])) {
          unset($lis[$cnt]->{0});
          break;
    }
  }
}
```

Ukázka kódu 13: Vytváření vlastního RSS zdroje2

Kapitola 5

Testování

Aplikace běží na adrese `http://185.8.165.84/msc/`.

5.1 Filtrování článků

Po dobu tří dnů byly předmětem testů tři uživatelé. Každý z nich měl nastavený MSC dotaz a každý den, po dobu tří dnů, se každému automaticky těsně po 4:00 ranní vytvářely filtrované RSS zdroje. Nastavení bylo následující:

U1: Uživatel1 měl nastaven dotaz *20E18*.

U2: Uživatel2 měl nastaven dotaz *22E43*.

U3: Uživatel3 měl nastaven dotaz *60G55,83C45,05C83*.

Výsledkem testu měla být úspěšnost filtrujícího algoritmu, a tedy schopnost správně vyhodnotit relevanci článku. Výsledná úspěšnost byla 100 %.

5.2 Založení uživatele

Předmětem testů se stal i proces registrace nového uživatele a následná úspěšnost akcí možných s registrovaným účtem. Testováním prošly formuláře pro registraci, přihlášení, zapomenuté heslo a změnu hesla. Autora zajímala především funkčnost těchto akcí, ale také emailové notifikace, které akce registrace a zapomenuté heslo generovaly. Uspěšnost obdržení notifikací byla 100 %, aniž by notifikace spadly do spamu. Všechny formuláře také poskytovaly implementované služby v plném rozsahu.

5.3 Čas běhu

Při sledování stejných uživatelů jako v první sekci této kapitoly byla otestována i časová náročnost aplikace. Při třídenním testování bylo zjištěno, že

v závislosti na velikosti zdrojového RSS souboru, trvá jeho stažení a uložení mezi 7 a 22 vteřinami. Sledované časové rozpětí je poměrně velké z důvodu velkého rozpětí počtů článků obsažených v daných RSS zdrojích. Počty článků se pohybovaly mezi 182 a 342.

S časem běhu souvisí i délka stahování metadat jednotlivých článků z původního zdroje. Přístup ke všem článkům přes ArXiv API, popisovaném v podsekci 2.4.1.1, zabral mezi 102 a 200 vteřinami. Do uvedených časů jsou započítána případná zdržení, kdy aplikace musí stahování nebo jinou akci (například přístup do databáze) opakovat kvůli neúspěšnému pokusu.

5.4 Míra označení MSC kódy

Během testování autor také zjistil, ze dvou náhodných po soubě jdoucích RSS zdrojů[18] (dny 16.5.2016 a 17.5.2016), které obsahovaly celkem 524 článků, že poměr článků, které mají MSC klasifikaci zadanou pouze ve zdrojovém PDF souboru, ku počtu všech článků je přibližně 11 %. Poměr článků, které nemají vyplněnou MSC klasifikaci na jejich příslušné HTML stránce, ku počtu všech článků je pak přibližně 53%.

5.5 Řešení chyb

Velmi důležité bylo zajistit automatickou funkčnost spouštění aplikace v 4:00 bez dalšího přičinění. Z toho důvodů si aplikace musí umět poradit s případnými problémy například se stahováním souboru nebo zapisováním do databáze. Dané časté problémy byly ošetřeny pomocí návratových kódů funkcí a jejich využití pro indikaci ukončení `while` cyklů, do kterých jsou tyto úseky kódu obaleny.

[18] `http://export.arxiv.org/rss/math`

Závěr

V rámci práce autor vypracoval analýzu s důrazem na rozbor existujících řešení a popis zdrojů dat. V kapitole Analýza autor definoval základní pojmy pro práci, porovnal stávající řešení podobných problémů s vysvětlením jejich přístupů a popsal hierarchii a způsob uložení zdrojových dat pro aplikaci – článků. Analytická část v neposlední řadě představuje problém, který ovlivnil návrh a implementaci řešení, a to znemožnění automatizovaného stahování zdrojových PDF dokumentů.

Autor dále vytvořil návrh řešení s pomocí UML diagramů a výstupem v podobě prototypu. V kapitole Návrh také definoval funkční a nefunkční požadavky na aplikaci, které nakonec splnil, byť s omezeními danými problémem představeným v kapitole Analýza. Popis toho, nakolik a jakým způsobem práce tyto požadavky pokryla, je poskytnut v kapitole Implementace. V neposlední řadě autor představil i požadavky na uživatele aplikace.

Kapitola Implementace obsahuje popis použitých technologií a konkrétní ukázky řešení určitých úloh v rámci aplikace. Časté ukázky zdrojového kódu dokumentují praktickou část práce.

Cíl práce byl splněn, byť ne tak, jak by si autor práce představoval. Problém zkodumentovaný v kapitole Analýza znamená znemožnění automatizovaného přístupu k PDF dokumentům z archivu arXiv.org. Přístup k PDF dokumentům byl tedy nakonec implementován manuální, kdy uživatel aplikace, pro možnost rozboru PDF souborů aplikací, musí tyto dokumenty sám stáhnout a nahrát do určené složky. Aplikace dál již umí soubory ve správné složce automatizovaně parsovat. Zbytek cílů byl autorem implementován dle jeho představ.

Práce počítá s možným budoucím rozšířením především ohledně množiny dat, kterou by aplikace mohla filtrovat. Dále by v budoucnu mohl být vypracován návrh bohatšího grafického rozhraní s větší možností zapojení uživatele, a to například v podobě specifičtějšího zadání dotazů (kombinace metadat).

Literatura

[1] HOLZNER Steven a Jan ŠINDELÁŘ. *RSS: Automatické doručování obsahu vašich WWW stránek.* Brno: Computer Press, a.s., 2007. ISBN 978-80-251-1479-7

[2] *MSC2010 database* [online]. American Mathematical Society. [vid. 12.4.2015]. Dostupné z: http://www.ams.org/msc/msc2010.html

[3] *arXiv.org e-Print archive* [online]. Cornell University. [vid. 10.3.2015]. Dostupné z: http://arxiv.org/

[4] *Paginated Search for Web Applications.* [online]. MarkLogic Community [online]. MarkLogic Community, 2012 [cit. 2016-05-17]. Dostupné z: https://developer.marklogic.com/learn/2006-09-paginated-search

[5] *arXiv.org Search* [online]. Cornell University. [vid. 10.3.2015]. Dostupné z: http://arxiv.org/find

[6] *PDF Parser | PHP library to parse PDF files and extract elements like text.* [online]. Sébastien MALOT. [vid. 20.4.2015]. Dostupné z: http://www.pdfparser.org/

[7] *Front for the arXiv* [online]. Department of Mathematics, UC Davis. [vid. 21.3.2015]. Dostupné z: http://front.math.ucdavis.edu/

[8] *arXiv.org help - arXiv API* [online]. Cornell University. [vid. 22.3.2015]. Dostupné z: http://arxiv.org/help/api/index

[9] *arXiv.org help - Robots Beware* [online]. Cornell University. [vid. 22.3.2015]. Dostupné z: http://arxiv.org/help/robots

[10] *HTTP/1.1: Status Code Definitions.* W3.org [online]. MIT , ERCIM , Keio, Beihang: W3.org [cit. 2016-05-17]. Dostupné z: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

[11] *Using a Three-Tier Architecture Model.* MSDN [online]. Redmond, Washington: Microsoft, 2016 [cit. 2016-05-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms685068(v=vs.85).aspx

[12] *http://arxiv.org/help/bulk_data_s3* [online]. Cornell University. [vid. 22.3.2015]. Dostupné arXiv.org help - arXiv Bulk Data Access - Amazon S3

[13] *A short proof of smooth implies flat* ArXiv.org e-Print archive [online]. Cornell University. [vid. 22.4.2015]. Dostupné z: http://arxiv.org/abs/1504.05709

A short proof of smooth implies flat. ArXiv.org e-Print archive [online]. Cornell University: Cornell University Library, 2015 [cit. 2015-04-22]. Dostupné z: http://arxiv.org/abs/1504.05709

[14] *Extensible Markup Language (XML)* [online]. MIT , ERCIM , Keio, Beihang: W3C. [vid. 25.4.2015]. Dostupné z: http://www.w3.org/XML/

[15] *Extensible Markup Language (XML) 1.0 (Fifth Edition)* [online].MIT , ERCIM , Keio, Beihang: W3C. [vid. 25.4.2015]. Dostupné z: http://www.w3.org/TR/2008/REC-xml-20081126/

[16] *Extensible Markup Language (XML) 1.1 (Second Edition)* [online]. MIT , ERCIM , Keio, Beihang: W3C. [vid. 25.4.2015]. Dostupné z: http://www.w3.org/TR/2006/REC-xml11-20060816/

[17] *RDF/XML Syntax Specification (Revised)* [online]. MIT , ERCIM , Keio, Beihang: W3C. [vid. 26.4.2015]. Dostupné z: http://www.w3.org/TR/REC-rdf-syntax/

[18] figure1.png (600x202). In: RDF/XML Syntax Specification (Revised) [online]. MIT , ERCIM , Keio, Beihang: W3C, 2004. [vid. 26.4.2015]. Dostupné z: http://www.w3.org/TR/REC-rdf-syntax/figure1.png

[19] MATULÍK Petr and Tomáš PITNER. Sémantický web a jeho technologie [online]. In: *Sémantický web a jeho technologie*. ÚVT MU, 2004. [vid. 27.4.2015]. Dostupné z: http://ics.muni.cz/bulletin/articles/296.html

[20] *RSS Tutorial for Content Publishers and Webmasters* [online]. Mark Nottingham. [vid. 27.4.2015]. Dostupné z: https://www.mnot.net/rss/tutorial/

[21] *W3C HTML* [online]. MIT , ERCIM , Keio, Beihang: W3C. [vid. 27.4.2015]. Dostupné z: http://www.w3.org/html/

[22] *Cascading Style Sheets* [online]. MIT , ERCIM , Keio, Beihang: W3C. [vid. 27.4.2015]. Dostupné z: http://www.w3.org/Style/CSS/

[23] *Why use CSS?* MDN [online]. MDN - Mozilla Development Network, 2016 [cit. 2016-05-17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/Why_use_CSS

[24] *PHP: What is PHP? - Manual* [online]. The PHP Group. [vid. 28.4.2015]. Dostupné z: http://php.net/manual/en/intro-whatis.php

[25] *ACID implementation in RDBMS.* Czech Technical University in Prague, 2011. Dostupné také z: https://users.fit.cvut.cz/valenta/doku/lib/exe/fetch.php/bivs/dbs-2/rdbms-architecutre-hadnout.pdf

[26] *Introduction — phpMyAdmin 4.5.0-dev documentation* [online]. The phpMyAdmin devel team. [vid. 28.4.2015]. Dostupné z: http://docs.phpmyadmin.net/en/latest/intro.html

[27] *Composer* [online]. Nils Adermann, Jordi Boggiano. [vid. 29.4.2015]. Dostupné z: https://getcomposer.org/doc/00-intro.md

[28] *Regular Expression Language - Quick Reference.* MSDN [online]. Redmond, Washington: Microsoft, 2016 [cit. 2016-05-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx

[29] *PHP: Introduction - Manual* [online]. The PHP Group. [vid. 27.4.2015]. Dostupné z: http://php.net/manual/en/intro.regex.php

[30] PTMFOG0000001530.png (1254x697). In: What is Cross Site Scripting and How Can You Fix it? [online]. Acunetix, 2015. [vid. 26.4.2015]. Dostupné z: http://www.acunetix.com/wp-content/uploads/2012/10/PTMFOG0000001530.png

[31] *Cross-site Scripting (XSS) Attack.* Acunetix [online]. Moorgate, London: Acunetix, 2016 [cit. 2016-05-17]. Dostupné z: http://www.acunetix.com/websitesecurity/cross-site-scripting/

[32] *HTTP Methods: GET vs. POST. W3 Schools* [online]. MIT , ERCIM , Keio, Beihang: W3.org, 2016 [cit. 2016-05-17]. Dostupné z: http://www.w3schools.com/tags/ref_httpmethods.asp

[33] *SQL Injection* [online]. MIT , ERCIM , Keio, Beihang: W3Schools.com. [vid. 10.5.2015]. Dostupné z: http://www.w3schools.com/sql/sql_injection.asp

[34] *PHP Prepared Statements* [online]. MIT , ERCIM , Keio, Beihang: W3Schools.com. [vid. 10.5.2015]. Dostupné z: http://www.w3schools.com/php/php_mysql_prepared-_statements.asp

Příloha **A**

Seznam použitých zkratek

HTML Hypertext markup language

XML Extensible markup language

RSS RDF Site Summary (pro RSS 1.0)

XSS Cross site scripting

UML Unified Modeling Language

SGML Standard Generalized Markup Language

RDF Resource Description Framework

URI Uniform Resource Identifier

CSS Cascading Style Sheets

PHP PHP: Hypertext Preprocessor

RDBMS Relation DataBase Management System

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `bp_frejisyn_bohdan_2015.pdf` ..... text práce ve formátu PDF