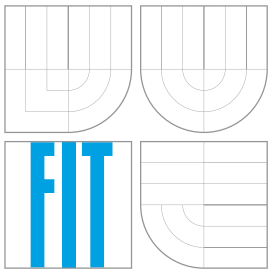
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

# BAKALÁŘSKÁ PRÁCE

Brno, 2016 Ondřej Šebela

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# AUTOMATICKÝ SYSTÉM PRO STAHOVÁNÍ VĚDECKÝCH ČLÁNKŮ

AUTOMATIC SYSTEM FOR DOWNLOADING OF SCIENTIFIC PAPERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ONDŘEJ ŠEBELA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV ROZMAN, Ph.D.
SUPERVISOR

BRNO 2016

# Abstrakt

Cílem práce je tvorba aplikace pro automatické stahování vědeckých článků s využitím internetových databází vědeckých publikací. Práce studuje strukturu a způsoby získávání informací o vědeckých dokumentech. Dále popisuje výslednou aplikaci a hodnotí ji na základě úspešnosti testování.



# Abstract

Goal of this work is the creation of application designed for automatic downloading of scientific papers by using online databases of scientific publications. This work is studying the structure and means of retrieving information about scientific documents. It also describes the final application and rates it based on results of testing.



# Klíčová slova

vědecké, články, Google, Scholar, Elsevier, Scopus, ScienceDirect, stahování, pdf, xml, json, aplikace, Django, Python, API, mashup, databáze

# Keywords

scientific, papers, Google, Scholar, Elsevier, Scopus, ScienceDirect, downloading, pdf, xml, json, application, Django, Python, API, mashup, database


# Citace

ŠEBELA, Ondřej. *Automatický systém pro stahování vědeckých článků.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.

# Automatický systém pro stahování vědeckých článků

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing.Jaroslava Rozmana Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Šebela
9. května 2016


## Poděkování

Děkuji vedoucímu práce panu Ing. Jaroslavu Rozmanovi, Ph.D. za pomoc a odborné rady při tvorbě této práce.

# Obsah

# Kapitola 1

# Úvod

Cílem práce je návrh a implementace aplikace umožňující vyhledání a stažení uživatelem zadaného vědeckého článku. Jednou z požadovaných vlastností aplikace je možnost vyhledávat další články rekurzivně podle citací obsažených v prvním zadaném dokumentu. Očekává se možnost omezení vyhledávání podle různých uživatelem zadaných kritérií. K činnosti aplikace měla využívat Google Scholar, ten však nevyhovoval díky striktním pravidlům pro jeho používání a omezoval tím práci aplikace. Aplikace proto namísto toho využívá Scopus a ScienceDirect, internetové databáze vědeckých publikací. Dalším požadavkem je snadné vyhledávání, zobrazování a filtrování již stažených dat a článků.

Jako srovnání mohou posloužit existující programy pro správu vědeckých článků, jako např. Papers, Docear nebo Paperpile[1] (využívá služby Google Docs). Tyto programy obsahují kromě vyhledávácích a třídících funkcí služby jako vkládání poznámek do PDF dokumentů, tvorbu citací dle zadaného formátu, možnosti organizace dokumentů dle různých kriterií apod.

Následující kapitola rozebírá zadané téma, zejména pak strukturu dokumentů ve formátu PDF a XML, API pro Scopus a ScienceDirect, popisuje ostatní databáze sloužící ke stejnému nebo podobnému účelu, popisuje principy Open Access a věnuje se obecně práci s těmito databázemi a vývoji aplikací využívající tyto databáze. Třetí kapitola popisuje předběžný návrh řešení aplikace, konkrétně jak by aplikace měla zpracovávat příchozí data, jejich ukládání do vlastní databáze aplikace, a následně jak tyto data zobrazit uživateli. Další kapitola se pak zabývá implementací aplikace, použitými nástroji, architekturou a uživatelským rozhraním aplikace. Pátá kapitola popisuje způsob testování a hodnotí výsledky tohoto testování.

[1] Seznam dalších podobných programů je dostupný na https://en.wikipedia.org/w/index.php?title=Comparison_of_reference_management_software&oldid=719221848

# Kapitola 2

# Rozbor zadání

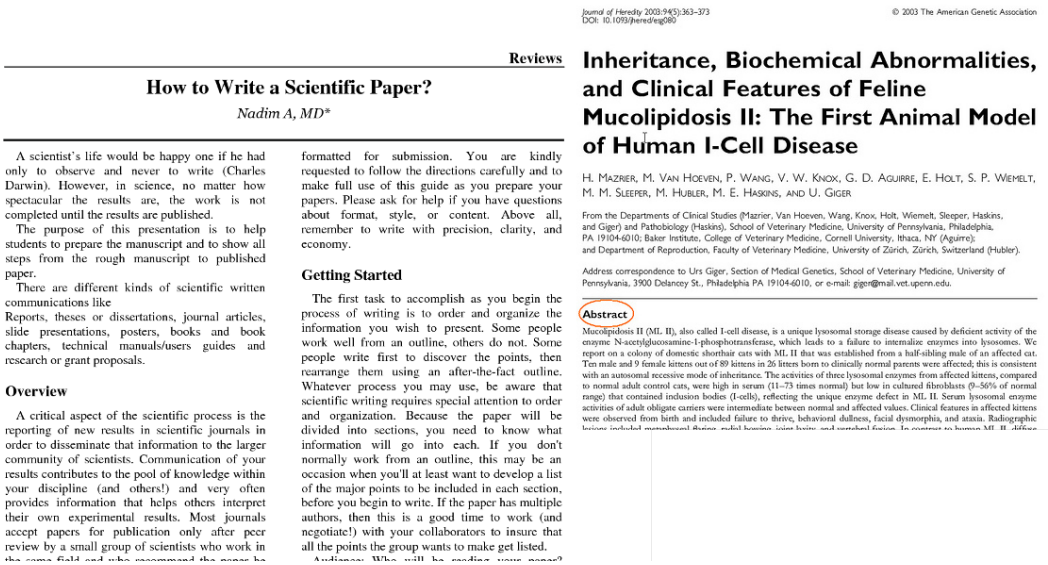
Tato kapitola popisuje teoretické znalosti a fakta potřebné k úspěšné realizaci aplikace. Popisuje definici vědeckého článku, strukturu PDF dokumentu, služby Scopus a ScienceDirect a jejich API.

## 2.1 Vědecký článek

Pro správnou funkci programu je potřeba definovat pojem vědecký článek, zvláště pak jeho strukturu.

### 2.1.1 Definice

Vědeckým článkem rozumíme formu vědecké publikace, jenž popisuje výsledky nějakého výzkumu, projektu nebo teorii. Jsou povětšinou psány odborníky na dané téma a před vydáním musí být přehodnoceny odborníky ze stejné oblasti [17]. Grafická úprava se může mírně lišit, ale struktura bývá často shodná.



Reviews

**How to Write a Scientific Paper?**

*Nadim A, MD**

A scientist's life would be happy one if he had only to observe and never to write (Charles Darwin). However, in science, no matter how spectacular the results are, the work is not completed until the results are published.

The purpose of this presentation is to help students to prepare the manuscript and to show all steps from the rough manuscript to published paper.

There are different kinds of scientific written communications like

Reports, theses or dissertations, journal articles, slide presentations, posters, books and book chapters, technical manuals/users guides and research or grant proposals.

**Overview**

A critical aspect of the scientific process is the reporting of new results in scientific journals in order to disseminate that information to the larger community of scientists. Communication of your results contributes to the pool of knowledge within your discipline (and others!) and very often provides information that helps others interpret their own experimental results. Most journals accept papers for publication only after peer review by a small group of scientists who work in

formatted for submission. You are kindly requested to follow the directions carefully and to make full use of this guide as you prepare your papers. Please ask for help if you have questions about format, style, or content. Above all, remember to write with precision, clarity, and economy.

**Getting Started**

The first task to accomplish as you begin the process of writing is to order and organize the information you wish to present. Some people work well from an outline, others do not. Some people write first to discover the points, then rearrange them using an after-the-fact outline. Whatever process you may use, be aware that scientific writing requires special attention to order and organization. Because the paper will be divided into sections, you need to know what information will go into each. If you don't normally work from an outline, this may be an occasion when you'll at least want to develop a list of the major points to be included in each section, before you begin to write. If the paper has multiple authors, then this is a good time to work (and negotiate!) with your collaborators to insure that all the points the group wants to make get listed.

Journal of Heredity 2003;94(5):363–373
DOI: 10.1093/jhered/esg080



**Inheritance, Biochemical Abnormalities, and Clinical Features of Feline Mucolipidosis II: The First Animal Model of Human I-Cell Disease**

H. Mazrier, M. Van Hoeven, P. Wang, V. W. Knox, G. D. Aguirre, E. Holt, S. P. Wiemelt, M. M. Sleeper, M. Hubler, M. E. Haskins, and U. Giger

From the Departments of Clinical Studies (Mazrier, Van Hoeven, Wang, Knox, Holt, Wiemelt, Sleeper, Haskins, and Giger) and Pathobiology (Haskins), School of Veterinary Medicine, University of Pennsylvania, Philadelphia, PA 19104-6010; Baker Institute, College of Veterinary Medicine, Cornell University, Ithaca, NY (Aguirre); and Department of Reproduction, Faculty of Veterinary Medicine, University of Zürich, Zürich, Switzerland (Hubler).

Address correspondence to Urs Giger, Section of Medical Genetics, School of Veterinary Medicine, University of Pennsylvania, 3900 Delancey St., Philadelphia PA 19104-6010, or e-mail: giger@mail.vet.upenn.edu.

**Abstract**

Mucolipidosis II (ML II), also called I-cell disease, is a unique lysosomal storage disease caused by deficient activity of the enzyme N-acetylglucosamine-1-phosphotransferase, which leads to a failure to internalize enzymes into lysosomes. We report on a colony of domestic shorthair cats with ML II that was established from a half-sibling male of an affected cat. Ten male and 9 female kittens out of 89 kittens in 26 litters born to clinically normal parents were affected; this is consistent with an autosomal recessive mode of inheritance. The activities of three lysosomal enzymes from affected kittens, compared to normal adult control cats, were high in serum (11–73 times normal) but low in cultured fibroblasts (9–56% of normal range) that contained inclusion bodies (I-cells), reflecting the unique enzyme defect in ML II. Serum lysosomal enzyme activities of adult obligate carriers were intermediate between normal and affected values. Clinical features in affected kittens were observed from birth and included failure to thrive, behavioral dullness, facial dysmorphia, and ataxia. Radiographic

Obrázek 2.1: Příklady dvou vědeckých článků. Převzato z http://ica.library.oregonstate.edu a http://www.gorskimd.com

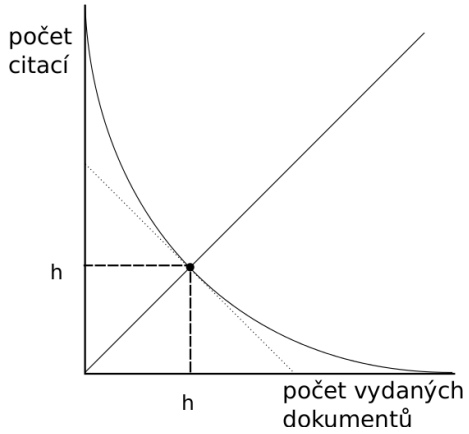
### 2.1.2 Struktura

Struktura většinou sestává z Názvu, Abstraktu, Úvodu, Metodologie, Výsledku, Diskuze a Citací. [17]

**Název** - název práce
**Abstrakt** - krátký popis práce
**Úvod** - další informace, zaměření práce
**Metodologie** - rozbor tématu, způsob jakým byl projekt vypracován, většinou popsáno tak, aby šlo pokus zopakovat
**Výsledek** - prezentace výsledků práce
**Diskuze** - zhodnocení a interpretace výsledků, ze kterých je vyvozen závěr. Může také obsahovat srovnání s předchozím nebo podobným výzkumem.
**Citace** - musí obsahovat seznam dokumentů na které se autor odkazuje v textu

### 2.1.3 Metriky článků

Google scholar i databáze Elsevieru vypočítávají a uchovávají různé metriky článků sloužící ke zobrazení viditelnosti a kvality článku na základě několika různých informací. Jednou z takových metrik je například h-index (Hirschův index[1]).




Obrázek 2.2: Graf h-indexu. Na ose x je znázorněn počet dokumentů vydaných autorem, osa y pak znázorňuje množství citací odkazujících se na dokumenty autora.

**H-index** autora je vypočten na základě počtu zveřejněných článků a počtu citací autorových článků. Problémem bývají rozdíly mezi obory, například jiné zvyky ohledně citování, nebo častější práce více autorů na jedné práci. Také může hrát roli dostupnost některých dokumentů. Tyto faktory mohou h-index ovlivnit a je tedy ho potřeba někdy upravit dalšími výpočty.

### 2.1.4 Identifikace článků

Při dnešním množství publikací na internetu vzniká potřeba jednoznačné identifikace jednotlivých dokumentů. Zatímco pro knihy vydané v tištěné podobě je zaveden identifikátor

[1] Popis dostupný na http://polymer.bu.edu/hes/articles/rp-hirsch05.pdf

**ISBN**, a pro periodika **ISSN**, digitální dokumenty lze identifikovat pomocí **DOI**, neboli **Digital Object Identifier**. Dokumenty vydané jak v tištěné tak digitální podobě mohou samozřejmě mít více těchto identifikátorů. DOI je spravován registrační agenturou CrossRef[2]. DOI má podobu URN a lze z něj vytvořit permanentní odkaz na dokument. Je složen z prefixu, který identifikuje organizaci, která publikaci vydala, a sufixu, který sestává ze zkratky názvu zdroje, jenž může být časopis, konference atd., roku vydání, ročníku publikace a interního identifikačního čísla zdroje. [15]

## 2.2 Formát PDF

PDF nebo *Portable Document Format* [9] je nejrozšířenější jazyk pro popis textu na stránce dokumentu. Byl vytvořen firmou Adobe v roce 1993 jako způsob sdílení dokumentů, který by byl nezávislý na platformě. Vychází z tehdy rozšířeného formátu PS (PostScript). Hlavní výhoda oproti formátu PS spočívá v tom, že není potřeba při zobrazování některé stránky z dokumentu není potřeba zpracovávat všechny předchozí stránky, což hlavně na tehdejších strojích významně zpomalovalo načítání dokumentů.

PDF lze zobrazovat pomocí programu Adobe Reader a vytvářet pomocí programu Adobe Acrobat, ale i pomocí jiných programů které podporují export ze jiného formátu do PDF, například Open Office. Roku 2008 byl vydán standard ISO32000-1:2008 který vychází PDF verze 1.7. Současně je ve vývoji verze PDF 2.0 pod standardem ISO32000-2.

### 2.2.1 Struktura dokumentu PDF

Struktura PDF byla tvořena s myšlenkou snadného a rychlého přístupu k jednotlivým stránkám dokumentu, a zároveň na co nejmenší velikost výsledného souboru. Na základě těchto požadavků má formát PDF objektovou strukturu, kde má každý objekt své jedinečné objektové číslo sloužící k identifikaci. Pomocí objektového čísla na sebe mohou objekty vzájemně odkazovat.

Soubor PDF sestává ze čtyř částí:
**Hlavička** (Header), která obsahuje informaci o verzi PDF souboru.
**Tělo** (Body) které obsahuje veškeré objekty v dokumentu.
**Tabulka křížových odkazů** (Cross-reference table) ve které se nachází informace sloužící k náhodnému přístupu k objektům.
**Trailer** umožňuje rychle najít aplikacím které čtou PDF dokument tabulku křížových odkazů ze které pak dále načítají objekty.

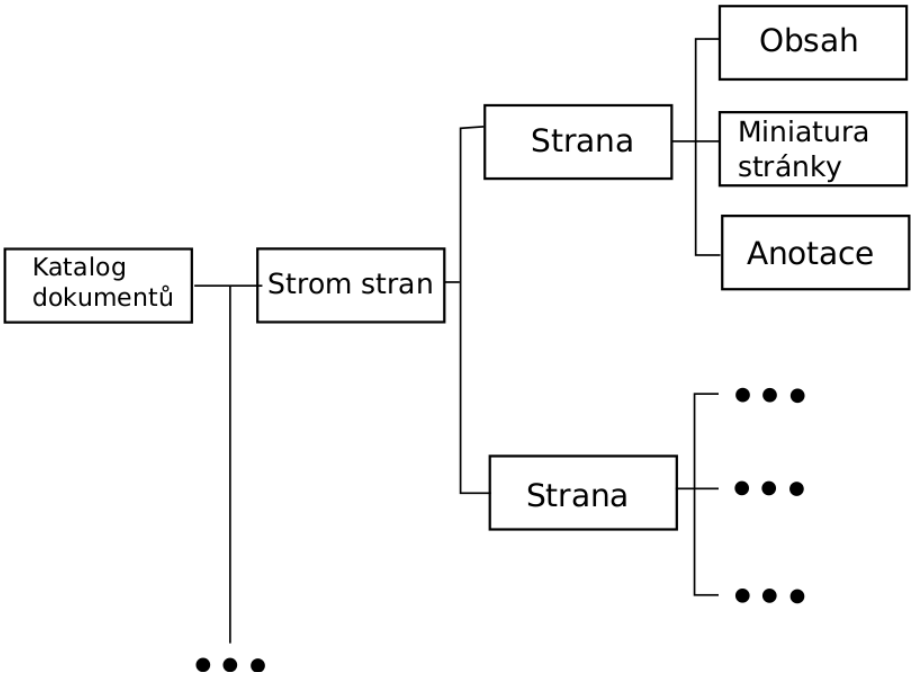
Objekty v těle soubory mohou být umístěny náhodně. Program který PDF soubor zpracovává načítá od konce, kde se nachází nejprve znak konce souboru, nad ním poté bytový offset poslední sekce křížového odkazu. O řádek výše je klíčové slovo *startxref* a nad ním se už nachází slovník sestávající z kličového slova *trailer* a párů hodnot *key* a *value*.

[2] Domovská stránka se nachází na http://www.crossref.org

2.1: Poslední řádky PDF souboru. Na začátku je slovník trailer s hodnotami, klíčové slovo *starxref*, offset poslední sekce křížového odkazu a znak konce souboru.

```
trailer
        key_1 value_1
        key_1 value_2
        ...
        key_n value_n
startxref
Byte_offset_of_last_cross-reference_section
\%\%EOF
```

Program zpracovávající PDF dokument musí nejprve zjistit, který z objektů je kořenový. Zjistí tak právě pomocí položek v sekci Trailer. Pomocí v něm uvedeného odkazu nalezne kořenový objekt, který je vždy typu katalog. Ten odkazuje mimo jiné na katalog stránek, který obsahuje stránky reprezentované objekty, uspořádané do vyváženého stromu.




Obrázek 2.3: Část hiearchie objektů v PDF. Kořenový objekt je typu katalog, ten obsahuje odkaz na objekt stromu stránek dokumentu, který obsahuje odkazy na jednotlivé objekty stran s obsahem, miniaturou strany a informacemi potřebnými k formátování textu.

Každý tento objekt obsahuje dva objekty. Jedním z nich je objekt s obsahem stránky. Tento objekt je typu stream, což znamená, že může obsahovat i binární data. Ty mohou být komprimovány ruznými algoritmy. Použitý algoritmus je obsažen ve slovníku objektu pod klíčem `/Filter`. Obsahuje pak hlavně text dokumentu, formátovací značky, určující velikost písma, jeho barvu, dále pak například i souřadnice znaků v textu. Může obsahovat i data obrázků obsažených v dokumentu.

Dále obsahuje objekt obsahující informace potřebné pro vykreslení stránky, mimo jiné například použité fonty, velikost písma, ale i odkazy na obrázky. Objekty které se v doku-

mentu nalézají vícekrát (např. obrázky), lze fyzicky uložit do jednoho objektu a pouze na něj opakovaně odkazovat a tím snížit velikost souboru.

### 2.2.2 Analýza textu dokumentu PDF

Text PDF dokumentu lze zpracovat použitím některé z dostupných knihovem. Krom oficiálního rozhraní dodávaných pod komerční licencí od firmy Adobe, jako je Adobe Reader nebo Adobe Acrobat, existuje mnoho volně dostupných knihoven a programů[3] schopných převádět text ze svého formátu do formátu PDF .

## 2.3 Formát XML

XML [14] je zkratkou pro Extensible Markup Language. Jedná se o značkovací jazyk, který byl vyvinut konsorciem W3C jako zjednodušená náhrada staršího jazyka SGML. Je využit pro serializaci dat, podobně jako JSON nebo YAML. Tím slouží k výměně dat mezi aplikacemi. Sám o sobě se jazyk nezabývá se grafickou reprezentací dat, může být však definována pomocí kaskádových stylů (CSS) podobně jako u HTML. Mezi hlavní výhody je dostupnost specifikace kterou poskytuje W3C, což usnadňuje implementaci do jakékoliv aplikace. Zpracování XML je schopna většina programovacích jazyků, ať už ve své základní podobě, nebo pomocí přidaných modulů.

### 2.3.1 Syntaxe XML

Dokument XML je text v kódování Unicode. Pro použitelnost XML dokumentu musí být kód strukturován určitým způsobem na základě následujících pravidel:

1. Dokument musí mít právě jeden kořenový uzel (angl. `element`).
2. Neprázdné uzly musí mít startovací a ukončovací značku (angl. `tag`), prázdné mohou být označeny speciální značkou.
3. Hodnoty atributů uzlů jsou opatřeny párem jednoduchých nebo dvojitých uvozovek. Tyto dva typy nelze kombinovat, lze však použít opačný typ uvozovek v hodnotě atributu.
4. Uzly lže vnořovat ale ne překrývat, tedy každý nekořenový uzel musí být celý obsažen v jiném uzlu.

XML rozlišuje velká a malá písmena v názvech uzlů, je tedy třeba dodržovat stejnou velikost písmen v počáteční a ukončovací značce uzlu.

Mezi dva nejrozšířenější způsoby zpracování XML dokumentu patří DOM (Document Object Model) parser, který převede XML do stromové struktury, a SAX (Single API for XML) parser, který dokument XML prochází a na jeho základě vyvolává události, které programátor dál zpracovává.

[3] Seznam takových programů lze nalézt na http://en.wikipedia.org/w/index.php?title=List_of_PDF_software&oldid=718901070

## 2.4 Formát JSON

JSON [16] neboli JavaScript Object Notation je způsob formátování textu určený pro výměnu dat. Lze snadno strojově číst i generovat. JSON je nezávislý na prostředí, jeho syntaxe je však podobná jazykům odvozených od jazyka C. Například jazyk Python při tisku seznamů nebo slovníků používá formát velmi blízký JSON jako textovou reprezentaci těchto struktur, a je schopen tyto struktury serializovat za pomocí svých standardních knihoven. Tento formát je hojně využíván ve webových aplikacích k serializaci dat, které je nutno zaslat JavaScriptu na uživatelské rozhraní.

### 2.4.1 Syntaxe JSON

Formát JSON rozlišuje dvě datové struktury, první typ `object` je kolekcí párů název - hodnota, a typ `array` je seznam hodnot. První jmenovanou strukturou lze reprezentovat typy jiných jazyků jako je slovník nebo typ `struct` v C, druhá reprezentuje datové struktury typu pole, seznam nebo vektor. Hodnoty v JSON jsou rozlišeny pouze jako typy `string` (řetězec) a `number` (číslo). JSON pak akceptuje i hodnoty `true`, `false` a `null`, a na místě hodnoty se může nacházet další vnořená struktura typu `object` nebo `array`.

2.2: Příklad syntaxe JSON: Objekt s vnořenými datovými strukturami.

```
{ 'exampleArray' : [1,2,3,4,5,6],
        'exampleObject' : {
                'exampleString' : "Hello world!",
                'exampleNumber' : 42
                }
}
```

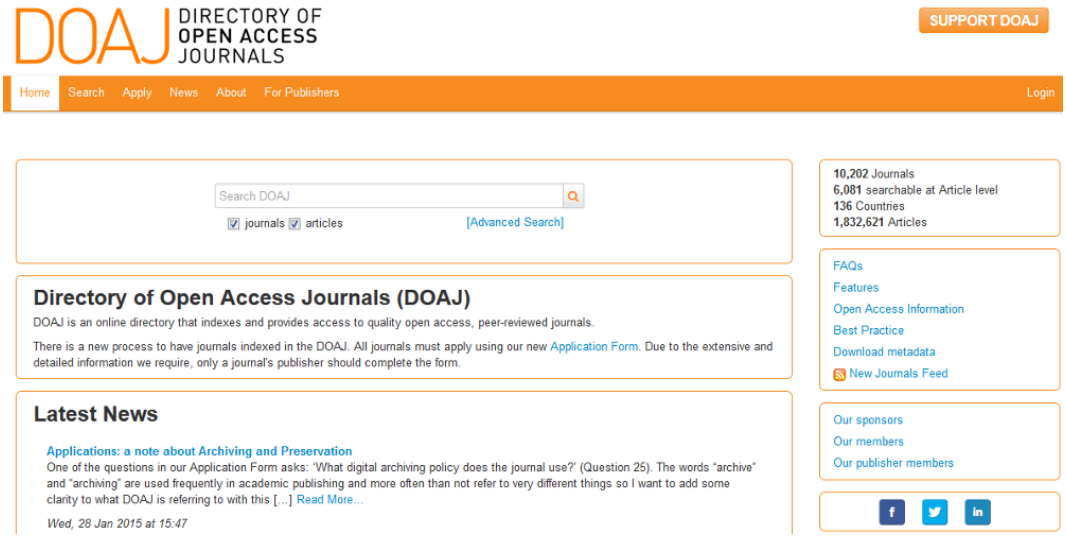
## 2.5 Otevřené databáze

Otevřené (angl. Open Access) databáze [8], jsou databáze volně dostupné uživatelům na internetu. Většinou se specializují na určitou oblast, časopisy, knihy nebo vědecké publikace, někdy i omezené na určité téma, např. přírodovědné nebo ekonomické publikace[4]. Umožňují uživatelům bezplatně vyhledávat a zobrazovat dokumenty které se v databázi nachází pomocí webového rozhraní. Cílem je umožnění volného přístupu k informacím především pro potřeby vzdělávání, princip OpenAccess lze však využít prakticky u jakéhokoliv média, včetně hudby nebo filmů.

### 2.5.1 Directory of Open Access Journals

Zkráceně DOAJ je webovou databází shromažďující otevřené časopisy, které jsou dostupné bezplatně všem čtenářům a institucím. DOAJ podporuje práva uživatelů na čtení, distrubuci, tisk, vyhledávání nebo citování těchto dokumentů[5]. Projekt je spravován švédskou Lundskou Univerzitou. Časopisy prochází před zařazením kontrolou. Musí splňovat principy Open Access, a musí mít přiřazeno identifikátor ISSN.

[4] Seznam několika OpenAccess zdrojů lze nalézt např. na https://www.vutbr.cz/knihovny/eiz/openaccess/zdroje
[5] Informace o projektu lze nalézt na http://doaj.org/about




Obrázek 2.4: Rozhraní webu Directory of Open Access Journals. Lze se zaregistrovat a na svém účtu shromažďovat nalezené časopisy.

Další podobnou službou je například Open Access Journals od Google, který taktéž shromažďuje časopisy, nebo Directory of Open Access Books a Google Books, které se zaměřují na knihy. Z českých například DML-CZ který shromažďuje významné české matematické publikace.

## 2.6 Google Scholar

Google Scholar [11] je internetový vyhledávač se zaměřením na vyhledávání vědeckých článků a prací. Byl poprvé spuštěn v roce 2004. Od roku 2006 lze importovat citace vytvořené v různých formátech, jedním z nich je BibTeX. Dále umožňuje uživatelům vyhledané výsledky shromažďovat do své vlastní knihovny. U vyhledaných článků lze zobrazit množství citací odkazující na právě zobrazovaný článek nebo zobrazit související články. U každého článku je veden záznam o počtu obsažených citací a jeho H-index. Google Scholar však nemá API a většinu automatizovaných skriptů efektivně blokuje, pro účely této práce se tedy ukázal jako nevhodný. Mezi alternativy patří například weby Scopus a Science Direct, nebo Web of Science, které jsou však placené. Zdarma je např. CiteSeerX který však spíše shromažďuje výsledky z jiných vyhledávačů.

Nevýhoda z pohledu programátorů je v případě Google Scholar neexistence API. Pravidla pro užívání databáze navíc zakazují opakované dotazy pomocí skriptů z důvodu nadměrného zatěžování systému. Takto odhalený skript je na určitou dobu zablokován a jsou odmítány veškeré jeho dotazy, což znemožňuje spolehlivou funkci takového programu.

## 2.7 Elsevier

Společnost Elsevier [1] existuje od roku 1880. Zaměřovala se na vydavatelskou činnost v oblasti vědy a medicíny. V dnešní době je součástí Reed Elsevier Group (zkráceně RELX) která se zabývá poskytováním informačních řešení pro oblasti vědy, medicíny ale i práv a

podnikání. Publikuje také různé časopisy a knižní tituly, a spolupracuje s předními vědeckými a zdravotnickými komunitami.

Společnost provozuje mnoho online informačních služeb, např. databáze Scopus a ScienceDirect, ale třeba i Reaxys, databázi zaměřující se na chemii, nebo GEOBASE, která poskytuje informace z oblasti geologie, ekologie, oceanografie a dalších souvisejicích oblastí[6].

### 2.7.1 Scopus

Scopus [4] je databáze citací a abstraktů akademické literatury, provozovaná společností Elsevier, která se zaměřuje na vydávání lékařské a vědecké literatury. Aby se předešlo střetu zájmů, je obsah databáze Scopus schvalován nezávislou radou odborníků. Databáze je plně přístupná pouze za poplatek. Scopus úzce souvisí s databází ScienceDirect, většina citací odkazuje právě na stránku citovaného článku na ScienceDirect.

### 2.7.2 ScienceDirect

ScienceDirect [3] je databáze provozovaná společností Elsevier zaměřená na shromažďování celých článků z různých oblastí vědecké literatury. Každý článek je v databázi uložen ve formátu PDF. Dostupnost PDF však může být podmíněna platbou. V databázi se však nachází i Open Access články dostupné komukoliv.

## 2.8 API

Zkratka API [10] znamená Application Programming Interface a obecně se jedná o soubor rutin, nástrojů a protokolů k sestavování softwaru a aplikací. API reprezentuje část software na základě jeho vstupů, výstupů, operací a datových typů se kterými pracuje. Umožňuje jednodušeji vytvářet nové programy, které využívají už existující software prostřednictvím API. API může být definována pro operační nebo databázový systém, může existovat jako knihovna pro práci s grafickými nástroji (např. DirectX nebo OpenGL), v neposlední řadě je pak využívání API rozšířeno v aplikacích na síti, kde existuje jako prostá definice vzdálených volání funkcí dané webové aplikace.

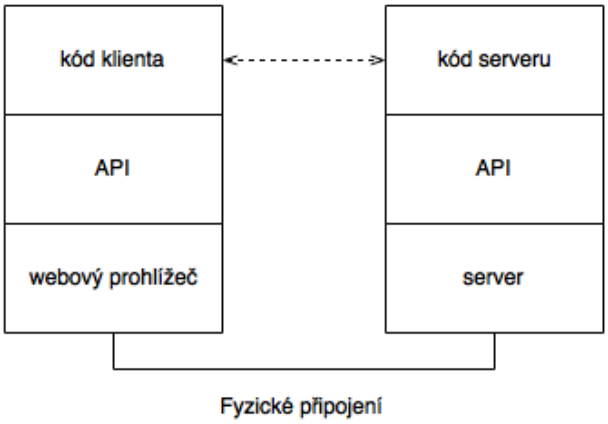
### 2.8.1 Web API

API síťových aplikací [13] je ve většině případů rozhraní, které na omezené množství HTTP dotazů odpovídá zprávami s předem definovanou strukturou, většinou ve formátu XML nebo JSON. API poskytuje tzv. koncové body. Ty jsou většinou definovány pomocí URI na kterou jsou zasílány HTTP dotazy, a ze které potom aplikace očekává odpověď od API. Tyto koncové body musí být statické aby byla zajištěna funkčnost programů které tyto koncové body využívají. Pokud dojde k větším změnám API, které vyžadují změnu funkcionality koncových bodů, je zvykem poskytovatelů API uchovat původní verzi API a jejích koncových bodů dostupnou po určitou dobu od zavedení nové verze. Tím se zajistí funkčnost softwaru, který API využívá a stále nebyl přizpůsoben nové verzi API.

### 2.8.2 Elsevier API

Databáze Scopus a ScienceDirect mají vlastní API. Přístup k API lze získat registrací, uživatel pak obdrží tzv. APIKey, tedy identifikační klíč k autorizaci programu, který na

[6] Celý seznam služeb Elsevieru lze najít na https:www.elsevier.comsolutions




Obrázek 2.5: Obecné schéma webové API.

API zasílá dotaz. Tento klíč je platný pro obě databáze. Na základě API klíče jsou rozlišeni platící a neplatící uživatelé. Neplatícím uživatelům nejsou přístupny všechny funkce API a není jim dovoleno zobrazovat text dokumentů, které nejsou šířeny pod Open Access licencí[7].

API poskytují několik koncových bodů, část je přístupná jen platícím uživatelům. V případě ScienceDirect stojí za zmínku hlavně „ScienceDirect Search API“, která umožňuje vyhledávání v obsahu databáze ScienceDirect na základě zadaného dotazu a definovaných kritérií. Jedno z kritérií může být například dostupnost článku pod licencí Open Access, datum vydání, země původu atd. Další zajímavý koncový bod je „Article retrieval API“, který na vyžádání vrátí celý text, nebo pouze abstrakt dokumentu, v závislosti na dostupnosti dokumentu pro daný účet. K dokumentu je potřeba znát jeho DOI, nebo jinou jedinečnou identifikaci (např. ISSN). Mezi další funkce API pak patří například získání metadat o dokumentu nebo vyhledávání objektů, které s ním souvisí[8]. API vrací odpověď ve formátu, který byl vyžádán pomocí parametru v dotazu. Lze si tak vyžádat data ve formátu XML nebo JSON. Díky možnosti získat v některém z těchto formátů i text dokumentu odpadá nutnost převádět PDF soubory do formátu, který lze snáze zpracovat.

V případě Scopus API je k dispozici funkce pro získání přehledu o citaci (Citation Overview API). Ta vrací souhrn metadat dokumentu, mezi nejdůležitější pro tuto práci patří seznam autorů a seznam institucí a vydavatelství, které dokument vydaly. Podrobnější data lze získat dotazem na konkrétního autora nebo organizaci pomocí Author Retrieval API a Affiliation Retrieval API, ty jsou však nedostupné pro běžné účty[9].

Hlavním problémem z pohledu této práce je nemožnost získávat informace o autorech a organizacích, lze pouze získat jména autorů, které jsou součástí citace článku. Dalším problémem je nedostupnost textu některých článků v databázi ScienceDirect. Lze však v API nastavit filtr, díky kterému API poskytne pouze Open Access články. Ty však mohou obsahovat citace článků ke kterým uživatel nemá přístup.

## 2.9 Mashup

Anglickým termínem mashup [12] se označují aplikace, které shromažďují data z více zdrojů, které pak zpracovávají a zobrazují je uživateli ve vlastním grafickém rozhraní. Často umož-

[7] Seznam dostupnosti funkcí lze získat zde: http://dev.elsevier.com/api_key_settings.html
[8] Celý seznam funkcí ScienceDirect API lze nalézt zde: http://dev.elsevier.com/sd_apis.html
[9] Celý seznam funkcí Scopus API lze nalézt na: http:dev.elsevier.comsc_apis.html

ňují data různě filtrovat, vizualizovat do grafů, seskupovat dle souvislostí a obecně s nimi pracovat jednodušším způsobem, než kdyby uživatel musel využívat původní zdroje dat odděleně. Mashup aplikace ve většině případů využívají API různých databází k získávání dat. Aby tyto aplikace měly zajištěn přístup ke svým datovým zdrojům na síti, jsou často implementovány jako webové aplikace.

## 2.10 Podmínky používání webových databází

Častým problémem při pokusu o automatizované získávání hromadného množství dat z webových databází bývají jejich omezení definovaná v podmínkách pro používání. Porušení takovýchto podmínek může mít za následek dočasné zablokování skriptu, ať už pomocí IP adresy, cookies, user-agentu nebo jiným způsobem. Ve vážnějších případech může následovat i trestní stíhání osoby odpovědné za takový skript. Nejčastějším problémem při získávání většího množství dat je příliš časté zasílání dotazů, které může být serverem vyhodnoceno jako chování neodpovídající lidskému.

### 2.10.1 Podmínky Google

Podmínky užívání společnosti Google, které jsou souhrnné pro všechny služby od Google (mezi které pochopitelně patří i Google Scholar), obsahují větu „Don't misuse our Services. For example, don't interfere with our Services or try to access them using a method other than the interface and the instructions that we provide.“[5] tedy „Nezneužívejte naše služby. Například, nezasahujte do chodu našich služeb nebo k nim nepřistupujte jinými metodami, než rozhraním a instrukcemi, které sami poskytujeme.“. Tato velmi široká definice tedy vylučuje jakékoliv čtení dat ze služeb Google pomocí skriptů. Navíc je soubor robots.txt [10] definován tak, že některé důležité odkazy a prvky stránky nezobrazuje, pokud byla stránka vyžádána nějakým programem. Při častých dotazech od takového skriptu pak Google Scholar vyžaduje vyplnění Captcha nebo všechny příchozí dotazy zablokuje na 24 hodin [7].

### 2.10.2 Podmínky Elsevier API

API pro servery Scopus a ScienceDirect nemá jasně daná omezení pro množství dotazů, mají však podmínky týkající se jasného označení zdrojů, ze kterých aplikace data získává. Je například potřeba uvádět zpětné odkazy na stránku dokumentu na ScienceDirect. Většina dalších pravidel pak přikazuje uvádět určitá data z API na stránce, nebo se týká webů, které svůj obsah nemají dostupný všem uživatelům [2].

[10] Dostupný na https://scholar.google.ca/robots.txt

# Kapitola 3

# Návrh

Vzhledem k nutnosti pracovat s online databázemi bude ideálním řešením vytvoření webové aplikace. Výsledná aplikace bude nejspíše zařaditelná do skupiny mashup aplikací, jelikož bude zpracovávat data z více zdrojů (Scopus, ScienceDirect) a ty prezentovat uživateli v určité jednotné formě ve vlastním grafickém rozhraní s vlastními funkcemi.

## 3.1 Požadavky

Na základě zadání by měl systém být schopen **vyhledávat a stahovat vědecké články** podle názvu článku, případně i pomocí autorů, organizace nebo vydavatele. Měl by umožňovat omezení vyhledávání tak, aby **nevyhledával články z určitého serveru**. Dále by měl zpracovávat pouze **dokumenty vyhledané pomocí služby Scopus a ScienceDirect**. V těchto vyhledaných dokumentech by pak měl najít citace odkazující na další články, a ty dále **rekurzivně stahovat**. Takto stažené související články by pak systém měl být schopen **vizualizovat**. V takto vytvořené databázi stažených článků by pak měl umožňovat **vyhledávat pomocí klíčových slov**.

## 3.2 Koncept

Systém bude sestávat z tří hlavních částí: **Vyhledávače, Analyzátoru textu článku** a **databáze** již stažených článků. Vyhledávač bude vyhledávat s pomocí služby Scopus požadovaný článek na internetu. Scopus vrátí výsledky vyhledávání, které aplikace zobrazí ve zjednodušené formě uživateli. Uživatel vybere článek, ten je pak vyhledán na ScienceDirect a je stažen ve formě PDF a XML. Po jeho stažení je článek zpracován analyzátorem, který z citací v XML přečte informace na základě kterých vyhledávač vyhledá další články, související s předchozím. Toto se děje rekurzivně dokud není dosaženo zadané hloubky vyhledávání. Pokud nelze některý z dokumentů vyhledat, nebo je jinak nedostupný (např. je potřeba zaplatit za přístup), je pouze zaznamenán jeho název a chybové hlášení a dál se nezpracovává. Pro všechny stažené dokumenty je vytvořen záznam v databázi, ten obsahuje krom informací o dokumentu a odkazů na záznamy autorů i cestu ke staženým souborům PDF a XML v souborovém systému. Závislosti jakéhokoliv článku bude možno vizualizovat grafem, který bude uživatel moci přibližovat, oddalovat a posouvat, případně bude moci aplikovat některý filtr aby vyloučil nechtěná data z grafu. Celý systém bude dostupný jako webová aplikace na serveru.

### 3.2.1 Vyhledávání

Vyhledávač bude spolupracovat s API databází Scopus a ScienceDirect. Při zadání dotazu uživatelem vyhledávač vytvoří dotaz na Scopus API, a jeho výsledky následně zpracuje a zobrazí uživateli. Uživatel vybere dokument ke zpracování a ten je pak odeslán analyzátoru a dále zpracován.

### 3.2.2 Zpracování dokumentu ve formátu PDF

Na obrázku 3.1 je znázorněn objekt dokumentu PDF, který obsahuje text. V ideálním případě takovýto objekt obsahuje text celé strany dokumentu, reálně je však text rozdělen do několika objektů, většinou podle kapitol a sekcí. Rozdělení mohou způsobit i vzhledové úpravy textu, například zvýraznění nebo nadpisy větším písmem mohou být v jiném objektu než zbytek textu. Extrémním případem může být i dokument, jehož objekty obsahují vždy po jednom znaku. Objekty v PDF se nemusí nacházet v takovém pořadí v jakém se zobrazí uživateli, a to díky operátorům transformace. Veškeré tyto případy by analyzátor měl být schopen rozpoznat a dokument správně načíst zpracovat.

3.1: Objekt v PDF dokumentu zobrazující text „Hello World“.

```
5 0 obj
<< /Length 67 >>
stream
BT
/F1 24 Tf
100 700 Td
(Hello World)Tj
ET
endstream
endobj
```

Načtení dat z dokumentu PDF bude ideálně řešeno pomocí některé z volně dostupných knihoven či webových služeb. Knihovna by měla umět PDF zpracovat a převést do formátu se kterým bude jednodušší dále pracovat. Jednou z takových možností je balíček PyPDF2 pro jazyk Python[6]. Ten umožňuje objekty PDF načítat jako objekty jazyka Python, číst pouze určené stránky apod. Pro jazyk C/C++ existuje knihovna Xpdf 3.0[7] která funguje i jako samostatná aplikace, nebo knihovna PoDoFo[8], která však není přímo uzpůsobená na extrakci textu z PDF. Pro PHP lze využít knihovnu PDFparser[9]. Dále je možnost zakoupit SDK přímo od Adobe.

Další možností je využití webových aplikací na převod PDF. Většina těchto aplikací se specializuje spíše na převod z různých formátů do formátu PDF než naopak. Jednou z aplikací, která umožňuje převod z PDF do jiného formátu je web Zamzar[10]. Umožňuje převod do textového formátu, ale i do html, nebo dokonce mp3. Přístup k API je však podmíněn

[6] Dostupný na https://pypi.python.org/pypi/PyPDF2/1.24
[7] Oficiální stránky http://www.foolabs.com/xpdf/
[8] Dostupné na sourceforge.net: http://podofo.sourceforge.net/
[9] Oficiální stránky: http://www.pdfparser.org/
[10] Dostupný na http://www.zamzar.com/

založením placeného účtu programátorem. Dalším webovým konvertorem je Cometdocs[11], který nabízí podobné služby jako Zamzar. Bohužel, je přístup k API omezen podobným způsobem. Třetím a pravděpodobně nejvhodnější webovou službou je pdfx[12]. Webová aplikace je zaměřena na vědecké články a převádí PDF do jazyka XML. Je volně dostupná bez jakýchkoli omezení.

Společným problémem většiny těchto aplikací je nespolehlivost při zpracování PDF které mají text rozdělen do 2 a více sloupců na stránce. Tento formát je zvlášť oblíben u zahraničních vědeckých publikací. Převaděče často spojí řádky ze dvou sloupců v jeden a text i citace se pak nedají dále správně zpracovat.

### 3.2.3 Zpracování dokumentu ve formátu XML

ScienceDirect API poskytuje na vyžádání celý text všech článků ve formátu XML. Tím odpadá potřeba převodu z PDF do programově čitelnějšího formátu, a s tím spojené nepřesnosti. API toto však umožňuje pouze u článků, ke kterým má majitel API klíče, kterým se aplikace autorizuje, přístup. V praxi to znamená, že je pro aplikaci dostupný text pouze u Open Access článků a článků, ke kterým má majitel aplikace zaplacený přístup.

[11] http://www.cometdocs.com/
[12] http://pdfx.cs.man.ac.uk/

3.2: Příklad citace v XML z textu článku na ScienceDirect

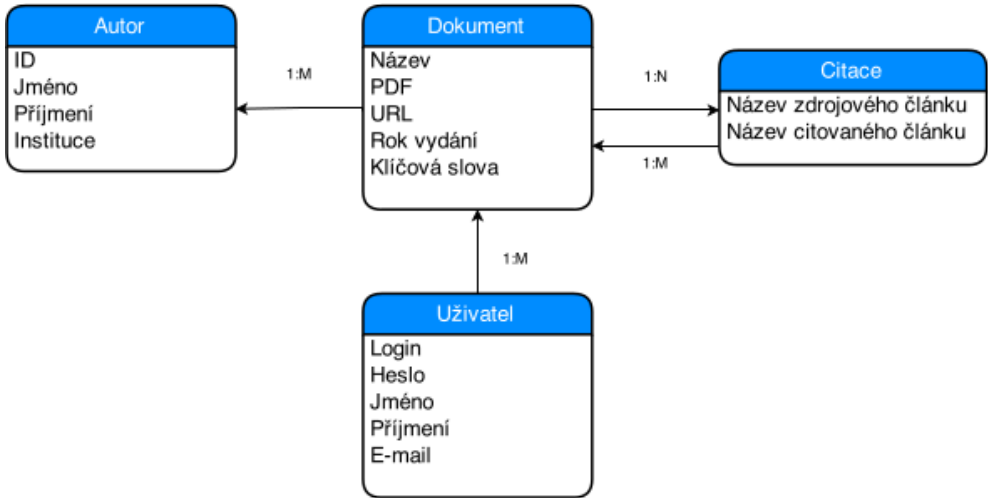
```
<ce:bib-reference id="br0010">
        <ce:label>[0]</ce:label>
        <sb:reference id="bib123456789987654321123456789">
        <sb:contribution langtype="en">
        <sb:authors>
        <sb:author>
                <ce:given-name>Jmeno</ce:given-name>
                <ce:surname>Prijmeni</ce:surname>
        </sb:author>
        <sb:author>
                <ce:given-name>J.</ce:given-name>
        <ce:surname>Prijmeni</ce:surname>
        </sb:author>
        </sb:authors>
        </sb:contribution>
        <sb:host>
        <sb:issue>
        <sb:series>
        <sb:title>
                <sb:maintitle>Nazev</sb:maintitle>
        </sb:title>
        <sb:volume-nr>42</sb:volume-nr>
        </sb:series>
        <sb:date>2016</sb:date>
        </sb:issue>
        <sb:article-number>123456</sb:article-number>
        </sb:host>
        </sb:reference>
</ce:bib-reference>
```

Citace jsou v XML (3.2) označeny uzlem `<ce:bib-reference>` který obsahuje atribut `id` pro snadnější identifikaci. Tento uzel pak obsahuje uzel `<sb:reference>`, který už obsahuje samotnou citaci, rozdělenou do uzlů `<sb:contribution>` a `<sb:host>`. První jmenovaný uzel obsahuje seznam autorů, někdy však obsahuje i uzly `<sb:title>` a `<sb:maintitle>`, které obsahují název citovaného dokumentu. Pokud není název zde, je pak obsažen v rámci uzlu `<sb:host>` ve stejně pojmenovaných uzlech. Analyzátor XML by měl být schopen nalézt název dokumentu v každém případě a název potom poskytnout vyhledávači pro další zpracování.

### 3.2.4 Ukládání dokumentů

Dokumenty bude potřeba ukládat do databáze kvůli snadnější organizaci a možnosti zaznamenat provázanost dokumentů citacemi. Dále databáze může posloužit jako podklad pro vizualizaci. Struktura navrhované databáze je naznačena na obrázku 3.1. V tabulce *Dokument* jsou uloženy informace o staženém dokumentu, jeho název, autoři, odkaz atd. Tabulka

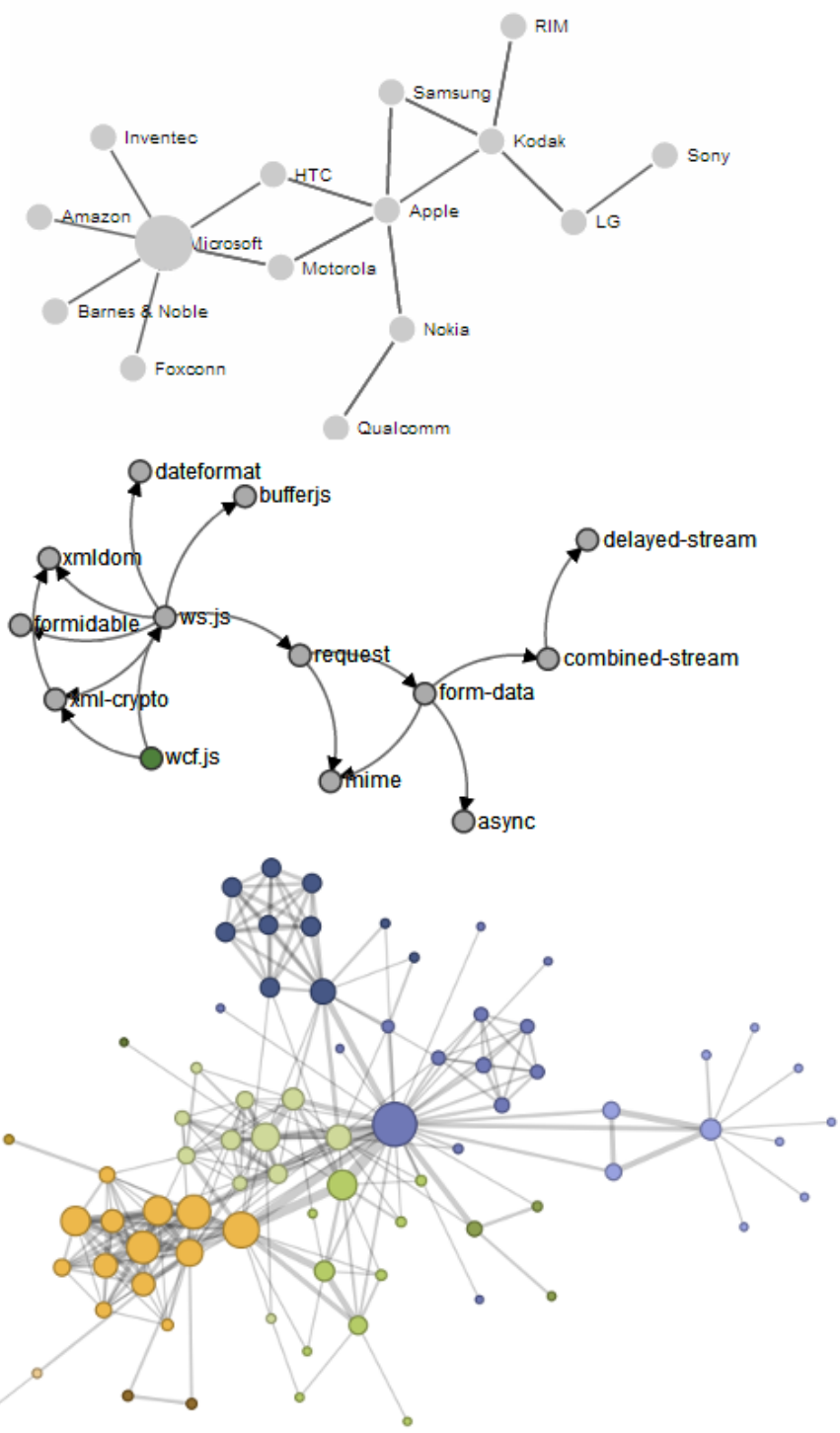
Autor bude obsahovat informace o autorovi, jeho příslušnost k určité instituci (univerzita, vydavatelství, atd.). Jeden dokument může mít více autorů, takže tabulka Dokument může obsahovat odkazy na více autorů. Tabulka Citace řeší M:N vztah mezi dokumenty. Na základě těchto informací bude možno závislosti článků vizualizovat. Tabulka uživatele eviduje všechny uživatele systému. Zajišťuje, aby uživatelé mohli pracovat s dokumenty, které si dříve vyhledali a stáhli.




Obrázek 3.1: Koncept databáze. Vlevo je tabulka s informacemi autorů, uprostřed tabulka dokument popisující jednotlivé dokumenty v databázi, vpravo tabulka citací díky které je možno znázorňovat závislosti mezi dokumenty, a dole tabulka uživatele.

### 3.2.5 Vizualizace

Uživatel bude mít možnost vizualizovat závislosti ke kterémukoliv staženému článku, který je uložen v databázi. Tento článek bude použit jako výchozí uzel grafu. Jelikož mohou citované články odkazovat i na jiné citované články, byla by stromová struktura nepřehledná [7]. Bude proto na místě využití grafu s uzly. Nejvhodnější nalezené grafy lze nalézt na obr. 3.2. Graf s orientovanými hranami a náhodným umístěním uzlů by přehledně vykresloval závislosti mezi dokumenty, avšak při větším počtu uzlů se stává nepřehledným. Úpravou vykreslovacího algoritmu lze umístění uzlů ovlivnit, vzniká pak graf řízený silou (force-oriented), který uzly sdružuje do skupin. V případě tohoto systému by byly v grafu skupiny tvořeny podle počtu vzájemných citací. Zároveň lze v takovém grafu zavést i možnost zvýrazňování uzlů podle důležitosti. Směrodatným údajem pro výpočet tohoto parametru by mohlo být množství článků které na dokument odkazují. Výsledkem by mohla být kombinace tří grafů na obrázku 3.2.

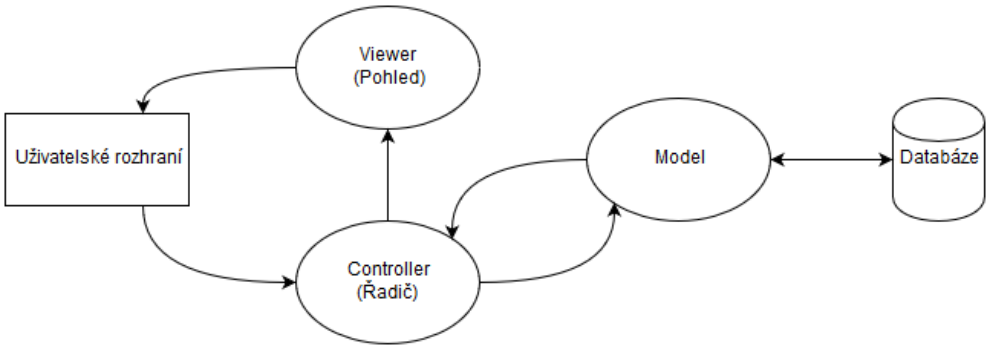



Obrázek 3.2: Příklady grafů. Nahoře jednoduchý graf s uzly, uprostřed graf s orientovanými hranami, dole silou řízený graf se zvýrazněním významných uzlů.

# Kapitola 4

# Implementace

K vytváření webových aplikací lze v dnešní době využít širokou nabídku vývojových prostředí, která jsou většinou založena na principu MVC (Model-Viewer-Controller). Mezi tyto možnosti patří například Ruby on Rails, Django framework založený na Pythonu, JavaServer Faces, nebo Nette Framework (PHP).




Obrázek 4.1: Schéma architektury Model-Viewer-Controller.

Základní myšlenka architektury MVC [6, Kapitola 9.2] je oddělení kódu logiky od výstupu. Tím se kód zpřehledňuje a díky tomu pak snáze udržuje a rozšiřuje.

**Model** v této architektuře obsahuje logiku, nezasahuje do výstupu ani k němu nemá přímý přístup. Pouze dostává data v parametrech a vrací zpět zpracovaná data na základě těchto parametrů. Zprostředkovává komunikaci s databází, většinou za pomoci ORM (Objektově-Relační mapování), kdy objekty jazyka odpovídají tabulkám v databázi.

**View** (Pohled) se stará o zobrazení uživatelského rozhraní. Ve webových aplikacích se nejčastěji jedná o šablonu v HTML, která je doplněna značkami některého značkovacího jazyka. Obsahuje jen minimum logiky, nejčastěji k ověřování validity dat a podobně. View neví odkud data přišla, pouze je zobrazuje uživateli.

**Controller** (Řadič, kontroler) propojuje Model a Viewer (Pohledy). Zpracovává vstupy od uživatele, které pak zasílá Modelu ke zpracování. Data přijatá od modelu pak zasílá do některého pohledu k zobrazení uživateli.

## 4.1 Django

Pro implementaci jsem zvolil framework Django, napsaný v jazyce Python. Jde o open-source framework určený pro webové aplikace, vytvořený neziskovou společností Django

Software Foundation. Využívá mírně upravenou architekturu MVC, kterou pojmenovává MTV (Model-Template-Viewer). Model je v zásadě stejný jako v architektuře MVC, Template (šablona) je obdobou pohledu v MVC, jde o HTML soubor se značkami („Template tags“) díky kterým lze vkládat některé základní konstrukce jazyka Python, a View (Pohled) je vlastně obdobou Controlleru (Řadiče), obyčejně však obsahuje více práce s daty než je obvyklé v architektuře MVC.

Objekty reprezentující tabulky databáze mají v Djangu širokou škálu funkcí pro vyhledávání a filtrování dat v databázi. Tyto funkce často využívají principu *lazy-evaluation*, není třeba se tedy příliš znepokojovat ohledně množství zaslaných dotazů na databázi. Ve frameworku jsou pak vestavěné funkce pro serializaci objektů reprezentující záznamy z databáze, které lze využít pro zasílání většího množství dat na uživatelské rozhraní.

K frameworku lze přidat jakýkoliv modul pro Python, většinou bez větších obtíží. K rozšíření funkcionality jsem využil moduly `BeautifulSoup` pro zpracování HTML a `Requests` pro zjednodušení zasílání HTML požadavků a jejich následné zpracování.

## 4.2 Vyhledávač

Vyhledávání začíná zadáním řetězce uživatelem do pole vyhledávání. Řetězec je zpracován a na jeho základě je vytvořen dotaz na Science Direct API. Jsou vyhledávány pouze Open Access dokumenty, aby byla zajištěna dostupnost potřebných dat pro další zpracování. Odpověď by měla sestávat z několika výsledků ve formátu XML, ty jsou zpracovány tak aby mohly být čitelně zobrazeny uživateli jako seznam dokumentů. V případě, že není nalezen žádný výsledek je uživatel informován.

Uživatel si ze seznamu vybere jeden dokument, ten je pak uložen do databáze a použit jako kořenový dokument, od kterého se pak odvíjí další vyhledávání a analýza. Při rekurzivním zpracovávání dokumentů vyhledávač vyhledává nové dokumenty na základě citací. Zde už není aplikován Open Access filtr, dokumenty které nelze získat jsou označeny jako nedostupné a nejsou dále zpracovány.

Získání informací o původu dokumentu, jako například instituce která dokument vydala, je realizováno pomocí dotazů na Scopus API, konkrétně vyhledáním pomocí DOI (Digital Object Identifier), které je obsaženo už v odpovědi ze Science Direct API. Poslední žádost je opět směrována na Science Direct API, a to kvůli získání XML verze celého dokumentu pro získání citací. Vyhledávač tedy pro každý dokument zašle tři dotazy, první na Science Direct API a druhý na Scopus API aby získal co nejvíce informací o dokumentu, třetí na Science Direct API pro získání citací.

## 4.3 Analýza dat

Data jsou částečně analyzovány už při prvotním vyhledávání. Je potřeba na základě odpovědi z API sestavit seznam dokumentů, k tomu je potřeba z odpovědi ve formátu XML přečíst základní informace o dokumentu, jako název, autory nebo datum vzniku, ale například i odkaz na zdroj PDF na Science Direct. K tomu je využíván modul ElementTree který by se nachází v základní instalaci Pythonu 3. Pomocí jeho funkcí je možné XML snadno analyzovat, a vyhledat uzly obsahující potřebné informace.

Jakmile uživatel zvolí zpracování jednoho z nalezených dokumentů, je dokument uložen do databáze a jsou přečteny citace které jsou v něm obsaženy. Citace jsou uloženy ve frontě, ze které program přečte citaci, předá k vyhledání. Pokud je dokument nalezen a je volně


Obrázek 4.2: Uživatelské rozhraní seznamu výsledků vyhledávání na databázi Science Direct.

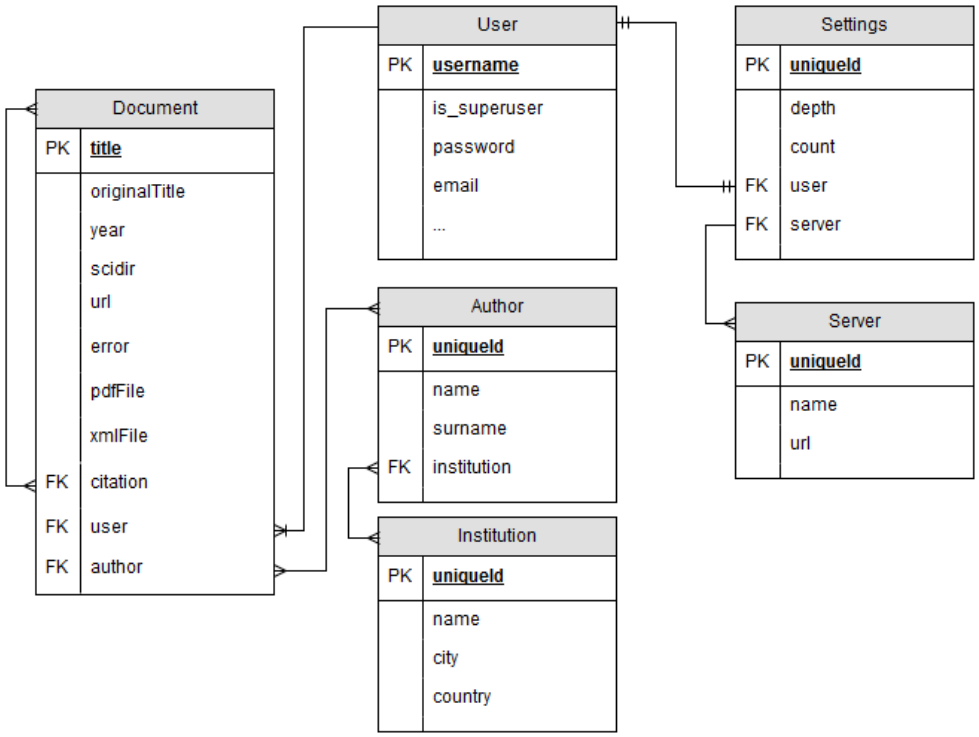
dostupný, je analyzován pro získání údajů o autorech, instituci, a je do databáze stažen ve formátu PDF. Poté jsou opět načteny citace v něm obsažené a proces se opakuje až do dosažení nastavené hloubky. Dokumenty které nejsou nalezeny, jsou nedostupné nebo obsahují chyby takového typu, že není možné je zpracovat, jsou označeny jako nezpracované a nejsou obsaženy v databázi. Zobrazí se pouze v konečném výpisu po dokončení celého procesu analýzy.

## 4.4 Databáze

Databáze je uložena ve formátu SQLite pro snadnou přenositelnost, je však možné v případě potřeby nastavit framework Django ke spolupráci s běžnou MySQL databází. Databáze obsahuje tabulky Document, Author, Institution, Settings a Server. Grafické znázornění databáze je na obrázku 4.3.

Document obsahuje informace o uloženém dokumentu a cesty k souborům PDF a XML v uložené v souborovém systému. Obsahuje také odkazy na zdroj dokumentu v databázi Science Direct. Tabulka je ve vztahu M:N sama se sebou a s tabulkou Author. Dále je ve vztahu 1:N s tabulkou User která je implementována přímo ve frameworku. Vztahy M:N jsou realizovány pomocí pole ManyToManyField, které vytváří tabulku obsahující páry primárních klíčů, ale navíc obsahuje funkce pro snadnější vyhledávání a práci s takto uloženými daty.

Tabulka Author obsahuje jméno a příjmení autora. Je v M:N vztahu s tabulkou Institution, opět realizovaným pomocí pole ManyToManyField. Tabulka Institution pak obsahuje informace o instituci, její název, město a zemi ve které se nachází.




Obrázek 4.3: Konečný způsob implementace databáze.

### 4.4.1 Uživatelský přístup

K aplikaci je nutné se přihlásit pomocí uživatelského jména a hesla. Účet může momentálně vytvořit kdokoliv. Lze díky tomu filtrovat dokumenty na základě uživatele který dokument nahrál. Uživatel má možnosti změny hesla, nastavení hloubky rekurze při práci analyzátoru, omezení vyhledávání na určité servery a smazání dokumentů které nahrál. Přihlašovací údaje uživatelů jsou uloženy v tabulce User kterou implementuje framework Django. Tabulky Settings a Server pak obsahují uložené nastavení pro uživatele. Jsou ve vztahu 1:N, tabulka Server ukládá názvy a adresy serverů, ze kterých uživatel zakázal stahování.

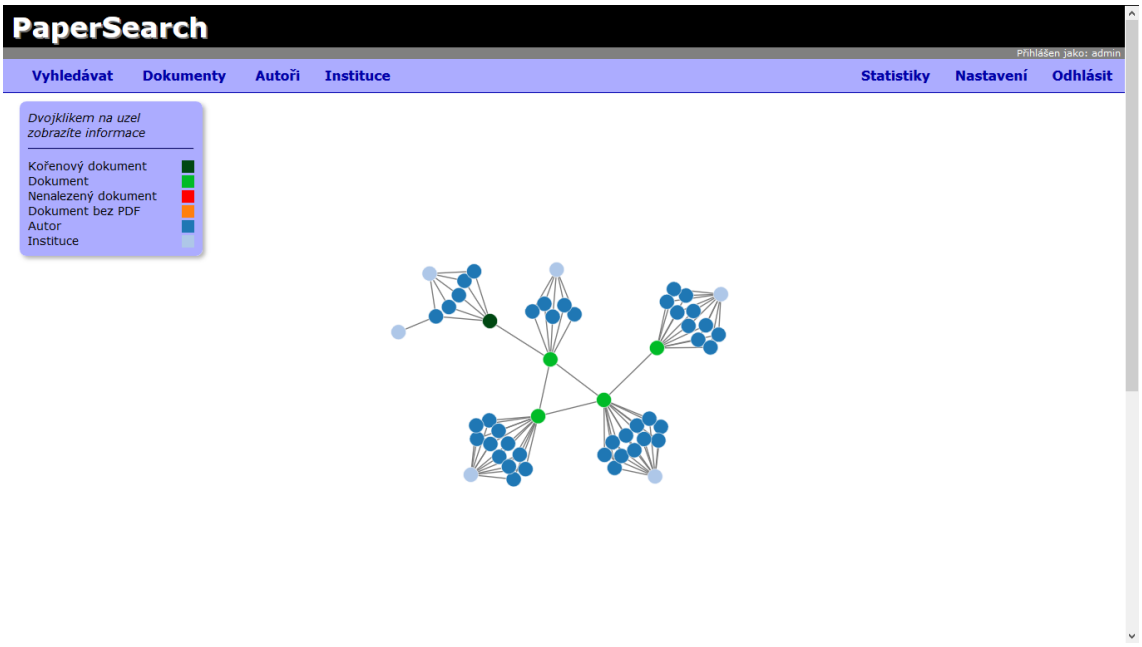
### 4.4.2 Zobrazování dat

Aplikace umožňuje zobrazení položek v tabulkách Document, Author a Institution. Jsou obsaženy v sekcích „Dokumenty“, „Autoři“ a „Instituce“ v uživatelském rozhraní. V každé sekci se pak zobrazí seznam položek i s informacemi a odkazy na související položky v jiných tabulkách, jako například položka Dokument, která obsahuje odkazy na autory dokumentu a ostatní dokumenty které cituje. Každá sekce obsahuje možnost vyhledávat podle zadaného textu, sekce s dokumenty pak umožňuje filtrovat dokumenty, u jejihž zpracování došlo k chybě, nebo je nahrál jiný uživatel. Oba tyto filtry lze libovolně vypínat a zapínat.

Každý uložený dokument pak nabízí možnost zobrazení vizualizace závislostí v grafu. Po zapnutí vizualizace je vytvořen silou řízený graf který zobrazuje veškeré dokumenty, které zvolený (kořenový) dokument cituje, k dokumentům jsou připojeny uzly reprezentující autory dokumentu, a k autorům pak instituce. Graf lze přibližovat i oddalovat a různě posouvat, pravým kliknutím na uzel lze zvýraznit pouze tento uzel a uzly, které s ním přímo sousedí. Dvojklik na uzel zobrazí podrobnější informace v levé části, v případě do-

kumentů i odkaz na PDF. Z grafu lze odfiltrovat uzly s autory, institucemi nebo neúspěšně zpracovanými dokumenty pro větší přehlednost.


Obrázek 4.4: Uživatelské rozhraní seznamu dokumentů, které byly nalezeny již uložené v databázi aplikace.



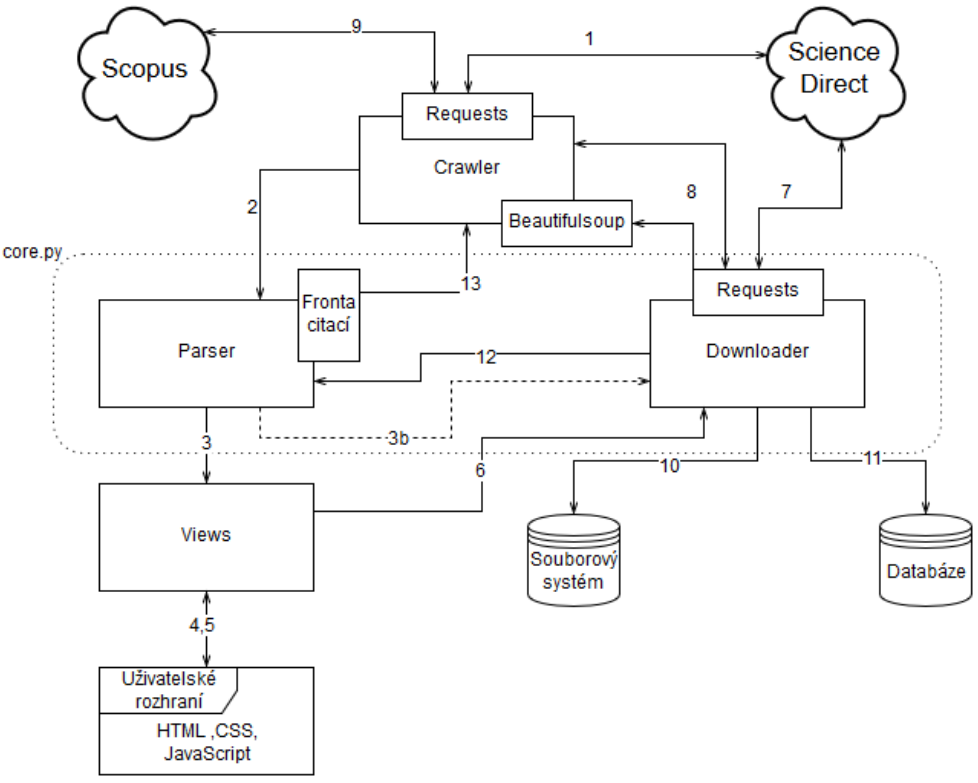

Obrázek 4.5: Vykreslený graf pro jeden z článků, ve kterém jsou odfiltrovány neúspěšně zpracované dokumenty.

### 4.4.3 Statistiky

Aplikace umožňuje zobrazení základních statistik v záložce statistiky. Na stránce se pak zobrazí počet dokumentů, autorů a institucí v databázi. V sekci dokumenty jsou pak i počty dokumentů, u kterých chybí PDF z jakéhokoliv důvodu, počet dokumentů, které nebyly dostupné pod licencí Open access a tedy nemohly být zpracovány, a počet dokumentů, které se nepodařilo zpracovat z jiných důvodů. Ke každé položce je vypočítáno i procentuální zastoupení v celkovém počtu.

## 4.5 Architektura aplikace

Výsledná aplikace je postavena na frameworku Django verze 1.9. Ke správnému běhu je potřeba modulů `BeautifulSoup4` a `Requests`. Uživatelské rozhraní je vytvořeno ručně pomocí HTML/CSS s vloženými příkazy frameworku Django, a pro jeho funkčnost je potřeba podpora JavaScriptu. Běh jádra aplikace zajišťují tři třídy. Třída `Crawler` zasílá s pomocí modulu `Requests` HTTP požadavky a zpracovává odpovědi ze serveru. Třída `Parser` pak zpracovává tyto odpovědi ve formátu XML, některé data zasílá na příslušný View, který data zpracuje pro zobrazení na uživatelské rozhraní, aby uživatel rozhodl co se bude zpracovávat. Úzce pak spolupracuje s třídou `Downloader`, ta opět využívá modul `Requests` ke komunikaci přes protokol HTTP, kdy si vyžadá PDF a XML soubory které pak společně s ostatními daty uloží do databáze.




Obrázek 4.6: Grafické znázornění architektury hlavní části aplikace.

1. Třída `Crawler` s pomocí `Requests` zasílá vyhledáváný řetězec zadaný uživatelem jako dotaz na Science Direct API.

2. Po obdržení odpovědi je odpověď ve formátu XML zaslána třídě `Parser`, která ji zpracuje.

3. Načtená data o nalezených dokumentech které odpovídají zadanému dotazu jsou zaslána na příslušný `View`.

4. Seznam nalezených dokumentů je zobrazen uživateli. Ten má možnost vybrat, který dokument se začne zpracovávat.

5. Po výběru dokumentu přečte `View` který dokument uživatel zvolil ke zpracování a zašle požadavek třídě `Downloader`.

6. Je potřeba ještě vyhledat dodatečné údaje o dokumentu a vyhledat odkaz na PDF verzi. `Downloader` zašle tyto požadavky třídě `Crawler`, k tomu jí pošle dříve uložený odkaz na stránku dokumentu na ScienceDirect, která standardně obsahuje odkaz na PDF verzi dokumentu, a DOI dokumentu.

7. `Crawler` zašle dotaz obsahující DOI na Scopus, ten vrátí další údaje, zejména informace o instituci, která dokument vydala.

8. Ze získaného odkazu stáhne `Downloader` PDF verzi dokumentu, vyžádá si i XML verzi pomocí API.

9. Oba soubory jsou uloženy na disk.

10. Jsou vytvořeny záznamy v databázi. Konkrétně záznam dokumentu se všemi získanými údaji a cestami k jeho PDF a XML souborům, záznamy autorů a záznamy institucí.

11. Odkaz na vytvořený záznam v databázi je zaslán `Parser`u, ten z XML verze dokumentu přečte veškeré citace a názvy titulů vloží je do fronty.

12. První titul je vyjmut z fronty a zaslán `Crawler`u.

Celý tento proces se pak opakuje dokud není dosaženo nastavené hloubky, akorát bez uživatelského vstupu, což eliminuje body 4,5 a 6. Namísto toho je rovnou zaslán požadavek třídě Downloader (bod 3b). Nenalezené nebo nedostupné dokumenty jsou do databáze uloženy s chybovým hlášením a dále se nezpracovávají. Jakmile je proces dokončen je zobrazen uživateli seznam dokumentů a jejich stav.

### 4.5.1 Zpracování dat pro vizualizaci

K vytvoření grafu jsem využil JavaScriptovou knihovnu `d3.js`. Ta vyžaduje data rozdělená na dva seznamy: `nodes` a `links`. Nodes reprezentuje seznam uzlů v grafu, jeho index v seznamu je využit i k jeho pozdější identifikaci, nešlo tedy využít identifikátory dané databázi. Seznam `Links` obsahuje informace o propojení uzlů, sestává pouze ze dvou identifikačních čísel, která reprezentují indexy uzlů v seznamu `Nodes`, a hodnotu `value`, která reprezentuje váhu tohoto propojení.

Aplikace tedy před samotnou vizualizací provádí převod dat z databáze na tyto dva seznamy v JSON. Tyto dva seznamy poté knihovna `d3.js` zpracovává tak, že z nich vytvoří dva seznamy objektů, které obsahují souřadnice v HTML prvku `svg` na stránce. Vlastnosti

těchto prvků jsou pak dynamicky upravovány JavaScriptem na základě událostí vyvolaných činností uživatele.

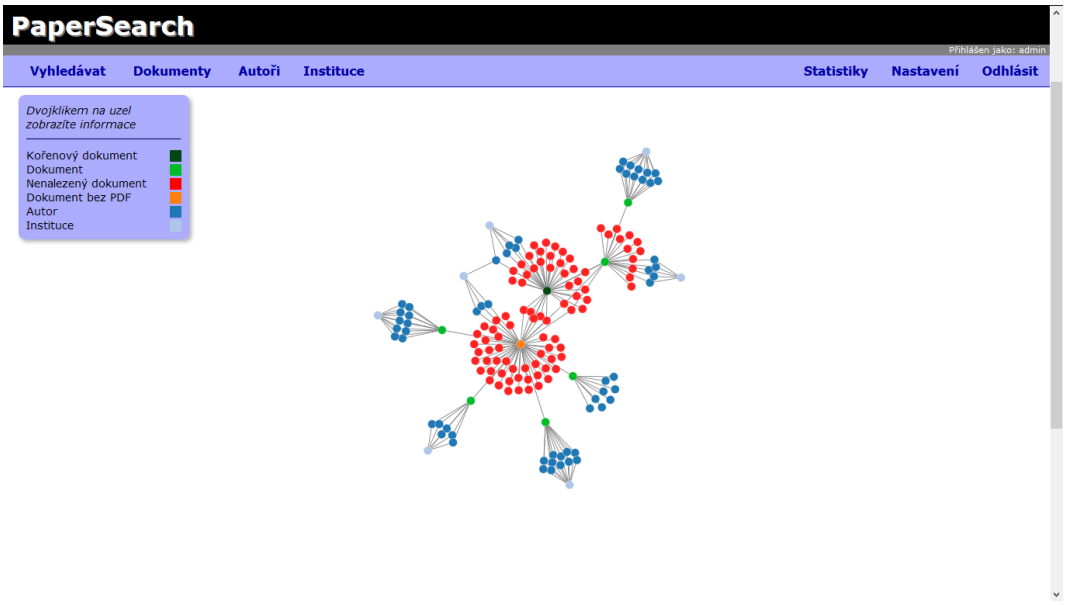
4.1: Zjednodušený příklad dat ve formátu JSON pro graf, sestávající ze dvou uzlů (dokumentů) a jednoho propojení.

```
{
    "nodes":[
        {
            "model":"app.document",
            "pk":"gravity-localization-on-hybrid-branes",
            "fields":{
                            // data z databaze
            }
        },
        {
            "model":"app.document",
            "pk":"appl-phys-lett",
            "fields":{
                            // data z databaze
            }
        }
     ]
    "links":[
        {
            "target":1,
            "value":"5",
            "source":0
        }
}
```

# Kapitola 5

# Testování

Při testování se ukázala hlavní nevýhoda databází ScienceDirect a Scopus. Díky tomu, že je k většině článků požadován přístup z placeného účtu, je většina dokumentů nedostupná. Při vyhledávání jsem takového články už odfiltrovával, aby alespoň počáteční (kořenový) dokument šlo zpracovat. Pokud však tento dokument obsahuje citace dokumentů, které nejsou na ScienceDirect přístupné pod licencí Open Access, nelze tyto citované dokumenty zpracovávat, protože jejich text, a tím i citace v nich obsažené, nelze získat. Tím se často analýza, která má probíhat například do hloubky 3, zastaví u některých dokumentů mnohem dříve.

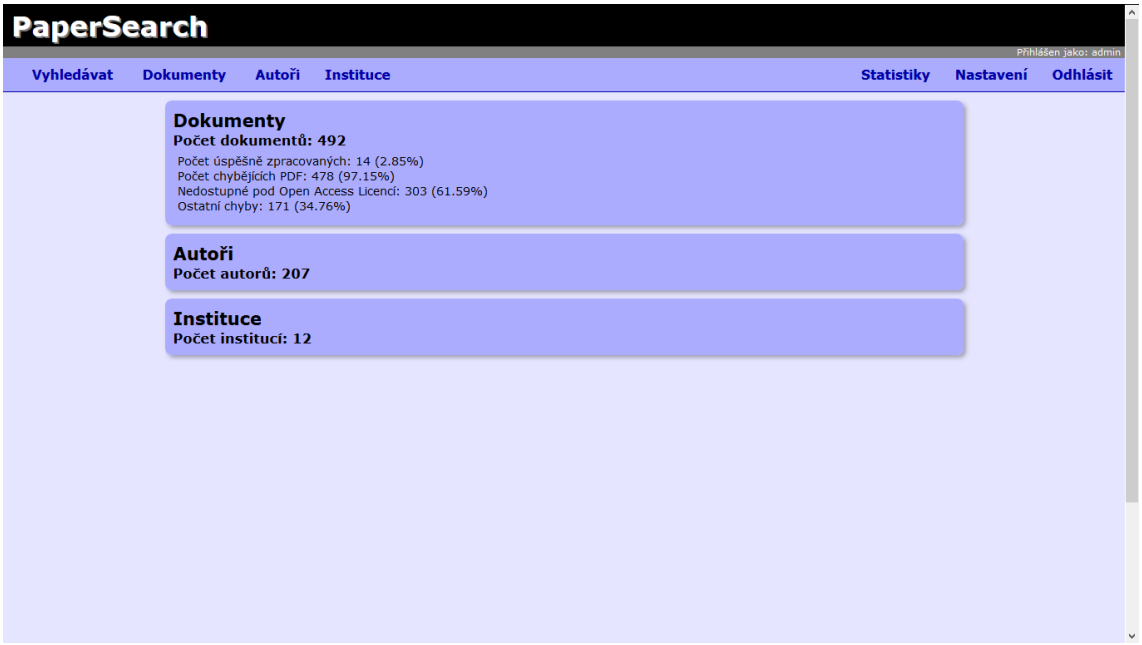



Obrázek 5.1: Graf závislostí vygenerovaný pro jeden z článků

Na obrázku 5.1 je jeden z grafů vygenerovaných pro článek z databáze. Lze si všimnout mnoha červených uzlů, které reprezentují dokumenty, které byly nedostupné pod licencí Open Access, nebo se nepodařilo je zpracovat z jiných důvodů. Oranžové uzly reprezentují dokumenty, které se podařilo zpracovat, ale stažení PDF selhalo. To znamená, že se sice podařilo získat ze Science Direct API verzi textu ve formátu XML, ale už ne verzi dokumentu v PDF. Odkaz na PDF nelze získat z API, takže ho program získává z webové stránky na Science Direct pro daný dokument. Jak jsem však zjistil, u některých dokumentů odkaz na

PDF zkrátka chybí.

K testování úspěšnosti jsem vyhledával obecné pojmy v angličtině z několika oborů, které zaručovaly alespoň jeden nalezený výsledek ke zpracování, např. pojmy „gravity“, „database“, „cancer treatment“ apod. Z nalezených dokumentů jsem vybral náhodně alespoň 1, a ten pak analyzoval s hloubkou nastavenou na hodnotu 3 bez dalších omezení. Takto jsem nechal analyzovat asi 8 dokumentů. Výsledný počet dokumentů v databázi byl 492.




Obrázek 5.2: Stránka zobrazující momentální statistiky

Na obrázku 5.2 jsou zobrazeny aplikací vypočtené statistiky podle záznamů v databázi. Podíl úspěšně zpracovaných dokumentů je velmi nízký, avšak 61.59% neúspěchů tvoří právě články, které nejsou dostupné pod licencí Open Access. Za předpokladu, že by se všechny tyto články podařilo korektně zpracovat, dosahovala by úspěšnost asi 64.43%. Pravděpodobně by však byla trochu nižší kvůli jiným faktorům při zpracovávání. Ostatní chyby při analýze byly většinou způsobeny naprostou nemožností najít citovaný dokument, nebo vyjímečně API odpověděla chybovým kódem, případně neposkytla korektní XML ke zpracování.

Nemožnost zpracovat většinu článků jsem se pokusil vyřešit zkontaktováním knihovny VUT a zajištěním přístupu na databáze článků přes školní autentizaci. Bohužel povaha aplikace přímo porušuje jeden z bodů používání školního přístupu[1], konkrétně zákaz hromadného stahování a dotazování za pomocí skriptů. Na základě tohoto pravidla jsem byl odmítnut, zástupce knihovny mě však informoval, že se pokusí zkontaktovat dodavatele databáze a zjistit jestli by to nešlo jinak. Bohužel jsem se odpovědi v dostatečném předstihu před odevzdáním práce už nedočkal.

[1] Všeobecné zásady používání školního přístupu na Scopus je možno nalézt zde https://www.vutbr.cz/knihovny/eiz/pripojeni

Aplikace samotná běžela bez větších problémů, nejdéle pochopitelně zabírá analýza dokumentů, délku běhu procesu pak ovlivňuje hlavně hloubka analýzy, jako vedlejší faktory lze předpokládat rychlost internetového připojení nebo výkon stroje, na kterém je aplikace spuštěna.

# Kapitola 6

# Závěr

Výsledkem této práce je aplikace, která je schopna vyhledat a stáhnout vědecké publikace, tyto dokumenty je pak schopna i s metadaty zobrazit uživateli v grafu. Během vývoje bylo třeba pozměnit po dohodě s vedoucím práce databázi se kterou aplikace bude spolupracovat, a to kvůli omezením, kterými Google Scholar znemožňoval korektní funkci programu. Aplikace tedy využívá databáze Scopus a ScienceDirect.

Potencionální úspešnost stahování je dostačující, avšak je momentálně omezena kvůli nemožnosti přístupu k většině článků, aplikace je kvůli tomu omezena pouze na dokumenty dostupné pod Open Access licencí. Účet s rozšířenými právy k zobrazování článků, který by tento problém eliminoval, se před dokončením práce nepodařilo získat.

Do budoucna je na prvním místě právě zajištění dostupnosti většiny článků. Kromě získání přístupu k dalším článkům z databází Scopus a ScienceDirect by bylo možné vyhledávát každý článek na více databázích a tím zajistit vyšší úspešnost. Dále by šlo aplikaci vylepšit o další funkce pro uživatele a upravit uživatelskou přívětivost grafického rozhraní. Za zmínku pak taky stojí systém vytváření a přihlašování uživatelů, který by zasloužil větší pozornost.

# Literatura


[1] Elsevier B.V.: About Elsevier. https://www.elsevier.com/about, [Online; Naposled zobrazeno 8.02.2016].

[2] Elsevier B.V.: Elsevier Developer Portal - API Policies. http://dev.elsevier.com/policy.html, [Online; Naposled zobrazeno 22.02.2016].

[3] Elsevier B.V.: ScienceDirect | Elsevier's leading information solution | Elsevier. https://www.elsevier.com/solutions/sciencedirect, [Online; Naposled zobrazeno 08.02.2016].

[4] Elsevier B.V.: Scopus | The largest database of peer-reviewed literature | Elsevier. https://www.elsevier.com/solutions/scopus, [Online; Naposled zobrazeno 8.02.2016].

[5] Google: Google Terms of Service – Privacy & Terms – Google. http://www.google.com/intl/en/policies/terms/, [Online; Naposled zobrazeno 22.02.2016].

[6] Grove, R.: *Web Based Application Development*. Jones & Bartlett Learning, 2009, ISBN 9781449661182.
URL https://books.google.cz/books?id=FBIsSbHANxAC

[7] Kobližek, O.: *Automatizovaný systém pro stahování vědeckých článků*. Bakalářská práce, Vysoké učení technické, Fakulta informačních technologií, 2012.

[8] Suber, P.: *Open Access*. MIT Press Essential Knowledge, MIT Press, 2012, ISBN 9780262300988.
URL https://books.google.cz/books?id=1wtmvu49g0YC

[9] Whitington, J.: *PDF Explained*. Oreilly and Associate Series, O'Reilly Media, 2011, ISBN 9781449310028.
URL https://books.google.cz/books?id=1TpDgwNGIK8C

[10] Wikipedia: Application programming interface — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Application%20programming%20interface&oldid=718858917, 2016, [Online; Naposled zobrazeno 19.04.2016].

[11] Wikipedia: Google Scholar — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Google%20Scholar&oldid=718943968, 2016, [Online; Naposled zobrazeno 8.02.2016].

[12] Wikipedia: Mashup (web application hybrid) — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Mashup%20(web%20application%20hybrid)&oldid=710631755, 2016, [Online; Naposled zobrazeno 20.04.2016].

[13] Wikipedia: Web API — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Web%20API&oldid=715934954, 2016, [Online; Naposled zobrazeno 19.04.2016].

[14] Wikipedia: XML — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=XML&oldid=717949842, 2016, [Online; Naposled zobrazeno 17.03.2016].

[15] WWW stránky: DOI Handbook. https://www.doi.org/hb.html, [Online; Naposled zobrazeno 05.02.2016].

[16] WWW stránky: JSON. http://www.json.org/, [Online; Naposled zobrazeno 17.03.2016].

[17] WWW stránky: Linnaeus University - Scientific Articles. http://lnu.se/the-university-library/search-and-writing-help-/scientific-articles?l=en, [Online; Naposled zobrazeno 22.01.2016].