Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Nástroj pro analýzu DBLP databáze

2024

Radek Vymětalík

Vedoucí práce:
RNDr. Martin Trnečka, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Radek Vymětalík
Název práce:	Nástroj pro analýzu DBLP databáze
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	RNDr. Martin Trnečka, Ph.D.
Počet stran:	47
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Radek Vymětalík
Title:	DBLP database analysis tool
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	RNDr. Martin Trnečka, Ph.D.
Page count:	47
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Bakalářská práce se zabývá vývojem webové aplikace pro analýzu DBLP databáze – přední databáze metadat publikací z různých oblastí informatiky. Práce nejprve seznamuje čtenáře s obsahem DBLP databáze a možnostmi jeho stažení. Následně popisuje použité technologie, motivaci pro jejich zvolení a implementační aspekty aplikace včetně její architektury a klíčových komponent. Ke konci práce je uveden postup zprovoznění aplikace a její uživatelská příručka.

Synopsis

The bachelor's thesis deals with the development of a web application for analyzing the DBLP database – a leading database of publication metadata from various fields of computer science. The thesis first introduces readers to the content of the DBLP database and its download options. Subsequently, it describes the technologies used, the motivation for their selection, and the implementation aspects of the application, including its architecture and key components. The end of the thesis provides instructions for setting up the application and its user guide.

Klíčová slova: webová aplikace; DBLP; vizualizace dat; Next.js

Keywords: web application; DBLP; data visualization; Next.js

Děkuji vedoucímu bakalářské práce RNDr. Martinu Trnečkovi, Ph.D. za jeho odborné vedení, čas a poskytnuté rady.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

DBLP Computer Science Bibliography, nebo jen DBLP, je přední veřejně přístupná databáze a webová služba poskytující metadata publikací z různých oblastí informatiky. Nalezneme v ní nejen publikace z mnoha významných časopisů a sborníků konferencí, ale i vědecké monografie a encyklopedie. V roce 2024 databáze obsahuje více než 7 milionů publikací od více než 3 milionů autorů z celého světa [1]. Nová data jsou do databáze neustále přidávána poloautomatickým procesem pod dohledem výzkumného centra Schloss Dagstuhl, které se snaží zaručit co nejvyšší kvalitu těchto dat.

DBLP poskytuje hodnotná data pro mnohé výzkumné pracovníky a instituce. Pro snadnější pochopení, srovnávání, sledování trendů a celkovou analýzu by však bylo vhodné mít možnost tato data a vztahy mezi nimi přehledně prezentovat. Právě k takovým účelům slouží online nástroj vytvořený v rámci této bakalářské práce. Pomocí různých typů grafů a tabulek přehledně prezentuje data stažená z databáze, především tedy metadata autorů, časopisů a konferencí. Nástroj umožňuje:

- vizualizovat statistiky autorů, časopisů nebo konferencí pomocí různých typů grafů,
- prohlížet metadata publikací s možností jejich filtrování,
- vytvářet skupiny autorů, prezentovat jejich statistiky a provádět nad nimi jednoduchou analýzu,
- vizualizovat grafy spoluautorství s možností filtrování zobrazených autorů podle různých parametrů,
- exportovat analyzovaná data.

2 Obsah DBLP

Všechen obsah DBLP databáze je přístupný skrze webovou službu, kterou nalezneme na adrese https://dblp.org/. Na této adrese jsou k dispozici jak webové stránky umožňující databázi pohodlně prohledávat, tak i volně dostupné webové API. Všechna data v databázi jsou zveřejněna pod licencí CC0 1.0 Public Domain Dedication license. Tato data tedy můžeme volně distribuovat, kopírovat, upravovat, přetvářet a vytvářet z nich odvozená díla, a to i pro komerční účely a bez vyžádání svolení [2].

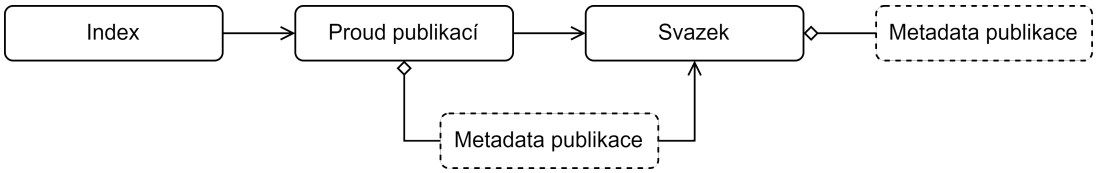
2.1 Základní typy stránek

Základní strukturu webové stránky tvoří pět typů stránek:

- stránky abecedně uspořádaných indexů,
- stránky proudů publikací,
- stránky svazků (obsahů),
- stránky jednotlivých publikací,
- stránky jednotlivých autorů.

Indexů nabízí webová stránka hned několik. Nejvýznamnějšími jsou index vědeckých časopisů, index konferencí a workshopů a index sérií. Série sdružují tématicky související publikace, třeba i z různých časopisů nebo konferencí. V těchto indexech jsou uvedeny odkazy na stránky *proudů publikací*, taktéž označovaných jako venues. Těmi se rozumí konkrétní vědecké časopisy, konference, workshopy nebo série. Na stránkách proudů publikací pak nalezneme odkazy na stránky *svazků*, které představují například jednotlivé svazky časopisů a konferencí nebo obsahy monografií.

Na stránkách svazků jsou uvedena metadata publikací. V mnoha případech jsou však metadata publikací uvedena již na stránce proudu publikací. Takové publikace většinou samy reprezentují stránky svazků a slouží jako odkazy na ně. Taktéž se může jednat o proudy publikací, které se již dále nedělí na svazky. Odkazy mezi zmíněnými stránkami a metadaty publikací můžeme vidět znázorněny na obrázku 1.

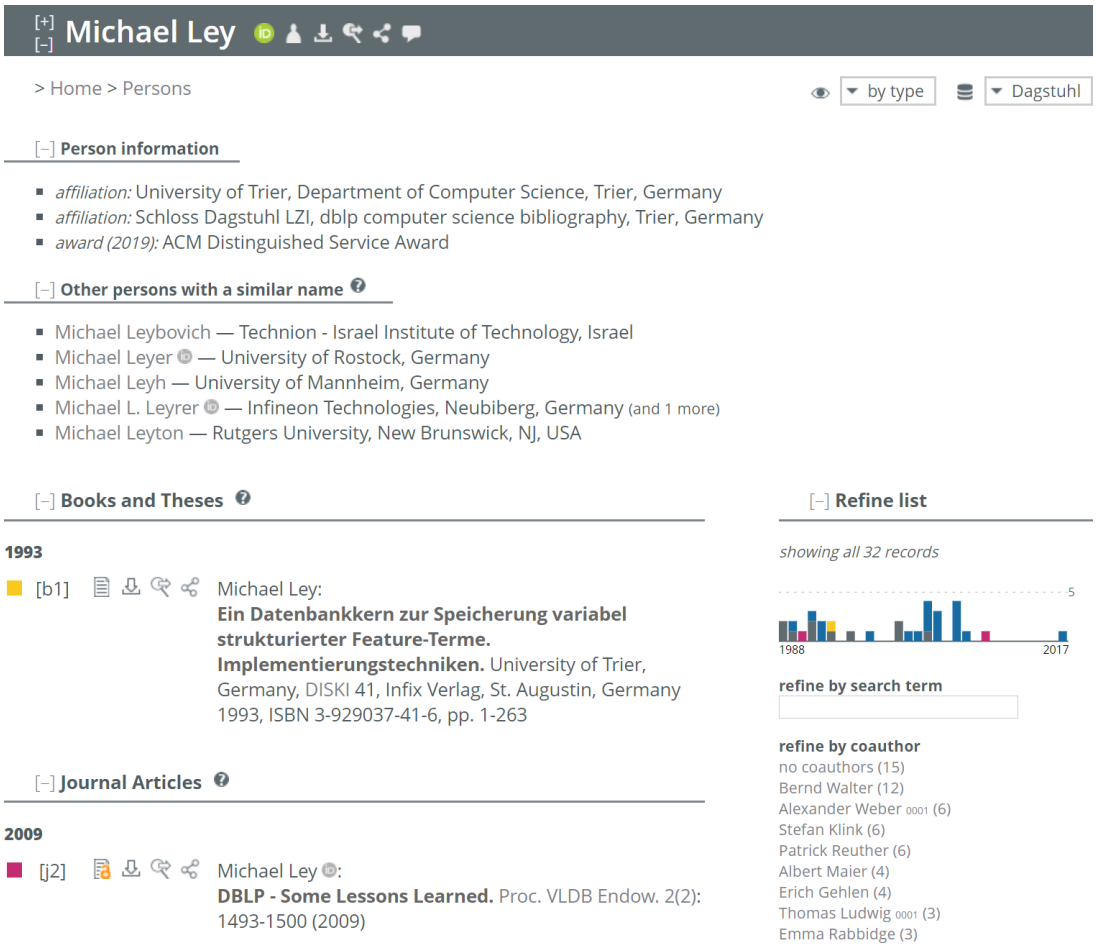


Obrázek 1: Odkazy mezi stránkami a metadaty publikací v DBLP

Všechna metadata publikace taktéž najdeme na samostatné stránce dané publikace. Na této stránce si lze navíc nechat načíst dodatečná metadata, která nejsou uložena v databázi, ze služeb OpenAlex [3], Crossref [4], OpenCitations [5] a Semantic Scholar [6]. Těmito metadaty jsou především:

- reference publikace,
- citace publikace,
- obsažené pojmy.

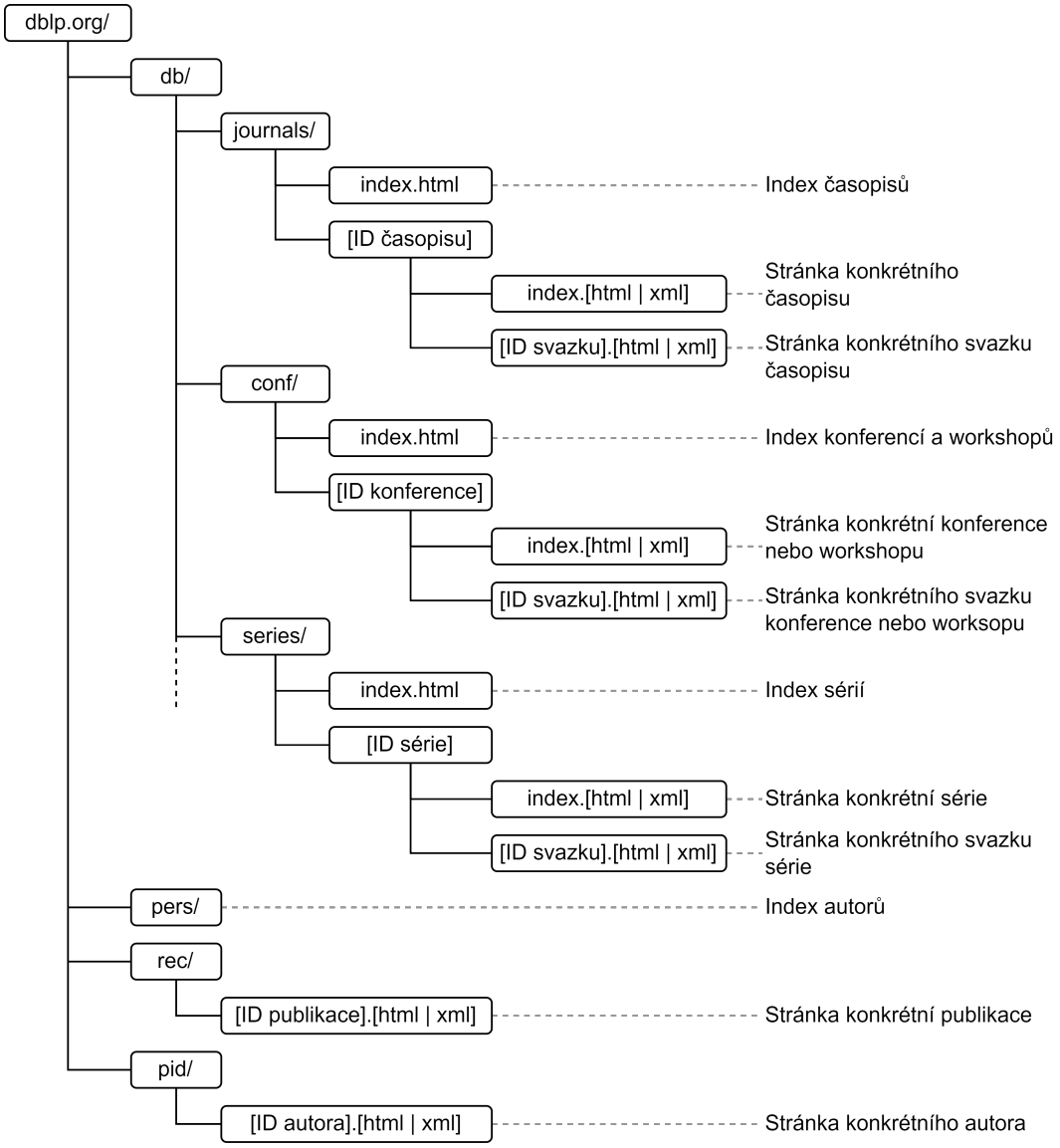
Všichni autoři uložení v databázi jsou dohledatelní skrze index autorů obsahující odkazy na stránky jednotlivých autorů. Na stránce autora jsou uvedeny jeho základní informace, jako například jméno, příslušnost k univerzitě nebo dosažená ocenění, a metadata všech jeho publikací nacházejících se v databázi. Na obrázku 2 můžeme vidět část stránky autora Michael Ley [7].



Obrázek 2: Část stránky autora Michael Ley v DBLP

2.2 URL adresy základních typů stránek

Na obrázku 3 můžeme vidět znázorněnu organizaci všech doposud zmíněných typů stránek z pohledu jejich URL adres. Kromě stránek indexů je obsah všech uvedených stránek k dispozici jak ve formátu HTML, tak i ve formátu XML. Identifikátor publikace, nebo též klíč, se skládá z několika segmentů. Úvodní segmenty typicky označují proud publikací, do kterého je publikace zařazena. Neplatí to však pro všechny publikace. Obdobně se i identifikátory autorů skládají z více segmentů.



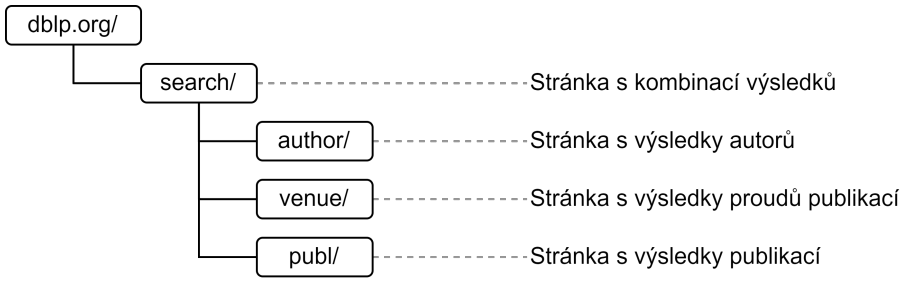
Obrázek 3: Struktura webové stránky DBLP

2.3 Vyhledávání

Pro snadnější prohledávání obsahu databáze poskytuje DBLP na svých webových stránkách vlastní vyhledávač založený na řešení jménem CompleteSearch [8]. Výsledkem vyhledávání je buď:

- seznam s odkazy na stránky autorů,
- seznam s odkazy na stránky proudů publikací,
- nebo seznam metadat publikací.

Ve výchozím nastavení vyhledávání zobrazí stránku s kombinací všech těchto typů výsledků. Je však možné si nechat zobrazit jen jeden určitý typ výsledku. Výsledek vyhledávání vždy obsahuje nejvýše prvních 1000 vyhovujících záznamů. Na obrázku 4 můžeme vidět organizaci stránek s výsledky vyhledávání z pohledu jejich URL adres.



Obrázek 4: Stránky s výsledky vyhledávání v DBLP

Ve výchozím nastavení vyhledávač porovnává každý hledaný výraz jako předponu potenciálního výsledku vyhledávání. Jestliže hledaný výraz obsahuje více slov oddělených prázdným znakem (například mezera nebo tabulátor) nebo znakem „+“, pak všechny musí být předponami v hledaném výsledku. Nerozlišují se velká a malá písmena a ani diakritika. Vyhledávání není fulltextové. Porovnávají se pouze některá metadata uložená v databázi, především tedy názvy publikací, proudů publikací a jména autorů. Možnosti konstrukce hledaných výrazů můžeme vidět v tabulce 1, kterou jsem převzal z FAQ sekce DBLP [9].

2.4 Vyhledávání pomocí API

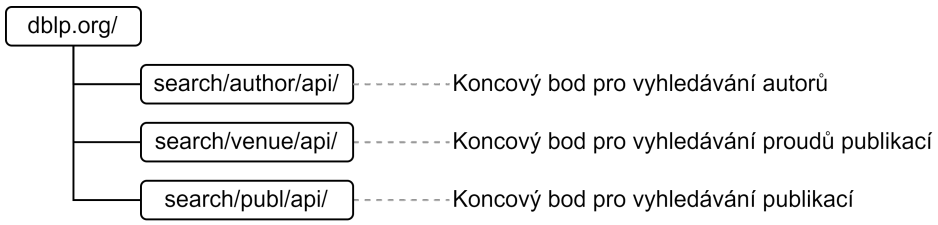
Stejné řešení vyhledávání, které lze použít na webových stránkách DBLP, je přístupné i skrze veřejné API. Toto API poskytuje tři koncové body pro vyhledávání autorů, publikací a proudů publikací. URL adresy těchto koncových bodů jsou znázorněny na obrázku 5. Výsledek vyhledávání můžeme ovlivnit několika parametry, jejichž výchozí hodnoty a příklady nalezneme v tabulce 2.

Hledaný výraz definujeme parametrem `q` URL adresy. Při konstrukci hledaného výrazu můžeme využít všechny možnosti uvedené v tabulce 1.

Tabulka 1: Možnosti konstrukce hledaných výrazů v DBLP

Možnost	**Syntax**	**Příklad**
prefixové hledání	výchozí	„sig“ se shoduje se slovem „SIGIR“ a také „signal“
přesné hledání	znak „$“ za hledaným výrazem	„graph$“ se shoduje se slovem „graph“, ale ne s „graphics“
booleovské AND	slova oddělená prázdným znakem nebo znakem „+“	„codd model“ se shoduje jak s výrazem „Codd's Relational Model“, tak i „A Model by Codd“
booleovské OR	slova oddělená znakem „|“	„graph|network“ se shoduje jak s „graph algorithm“, tak i „Network Analysis“

Parametrem `h` můžeme nastavit maximální povolený počet vrácených záznamů. V kombinaci s parametrem `f`, kterým definujeme hodnotu pořadí prvního vráceného prvku (pořadí je počítáno od nuly), je možné implementovat stránkování záznamů. Omezením tohoto API však je, že umožňuje vrátit najednou nejvýše 1000 záznamů, a to pouze z prvních nejvýše 10000 nalezených záznamů. K více než prvním 10000 vyhovujících záznamů se tedy nedostaneme.



Obrázek 5: URL adresy koncových bodů pro vyhledávání v DBLP

API vrací vedle konkrétních záznamů shodujících se s hledaným výrazem ještě i seznam celých slov, jejichž prefixem je slovo uvedené nejvíce napravo v hledaném výrazu. Jde tedy o slova, na která lze hledaný výraz doplnit. Maximální povolený počet takových slov, která API může vrátit, nastavíme parametrem `c`.

Formát vrácených dat definujeme parametrem `format`, který přijímá hodnoty:

- `xml` pro formát XML,
- `json` pro formát JSON,
- `jsonp` pro formát JSONP.

Tabulka 2: Parametry URL adres koncových bodů API pro vyhledávání v DBLP

Parametr	**Výchozí hodnota**	**Příklad**
`q`		`.../author/api?q=jan+novak`
`h`	`30`	`.../author/api?q=jan&h=200`
`f`	`0`	`.../author/api?q=jan&h=200&f=9000`
`c`	`10`	`.../author/api?q=jan&c=30`
`format`	`xml`	`.../author/api?q=jan&format=json`

2.5 Metadata publikací

Celý dataset s metadaty publikací je k dispozici ke stažení jako XML soubor `dblp.xml`, který je umístěný v adresáři na adrese https://dblp.org/xml/. Formát tohoto souboru je definován v DTD souboru `dblp.dtd`, který se nachází ve stejném adresáři. Stejný formát pro reprezentaci metadat publikací je použit i v XML souborech stránek proudů publikací, svazků, publikací a autorů. Detailnější informace o souboru `dblp.xml` a jeho formátu nalezneme ve FAQ sekci DBLP [10].

Metadata všech publikací vždy obsahují:

- unikátní identifikátor,
- název,
- typ publikace.

Volitelně (typicky v závislosti na typu publikace) pak metadata obsahují ještě například:

- seznam autorů nebo editorů,
- rok zveřejnění,
- název a identifikátor proudu publikací,
- název vydavatele,
- jeden nebo více odkazů na digitální vydání publikace,
- počet stránek.

DBLP uchovává metadata publikací několika typů. Typy publikací, které jsou prezentované uživateli a jejichž přehled můžeme vidět v tabulce 3, však nejsou úplně totožné s typy, pod kterými jsou metadata uložená v souboru `dblp.xml`. Metadata publikací jednotlivých typů, která se nachází v souboru `dblp.xml`, jsou inspirována typy záznamů formátu BibTeX. Konkrétně se jedná o následující typy záznamů:

- `article` – článek recenzovaného časopisu,
- `inproceedings` – příspěvek recenzované konference nebo workshopu,
- `proceedings` – svazek recenzované konference nebo workshopu,
- `book` – monografie,
- `incollection` – kapitola monografie,
- `phdthesis` – doktorská práce,
- `mastersthesis` – diplomová práce,
- `www` – webová stránka.

Kromě těchto typů autoři DBLP ještě zavedli vlastní typ `data` [11]. Tento typ je určen pro datasety a artefakty.

Tabulka 3: Typy publikací v DBLP

Anglické označení	**Český popis**
Books and Theses	monografie a vysokoškolské práce
Journal Articles	články, úvodní slova a předmluvy recenzovaných časopisů
Conference and Workshop Papers	příspěvky recenzovaných konferencí nebo workshopů
Parts in Books or Collections	výzkumné články, které byly vydány jako kapitola monografie
Editorship	editované publikace a sborníky
Reference Works	referenční materiály, výzkumné práce a encyklopedie
Data and Artifacts	datasety a artefakty (jako například software)
Informal and Other Publications	neformální a zbylé publikace typicky zveřejněné v online repositářích bez přísného prověřovacího procesu nebo na neformálních workshopech

2.6 Možnosti stažení dat

Z předchozích řádků plyne, že DBLP poskytuje několik možností, jak můžeme data z databáze stáhnout pro vlastní potřeby. Každá má své výhody i nevýhody:

- Stažení datasetu publikací – Jednou z možností je si stáhnout celý dataset publikací najednou jako jeden XML soubor. Zde je výhodou vysoká flexibilita. Nijak nás neomezuje podoba a vlastnosti API, které DBLP poskytuje, a můžeme s daty pracovat jakkoliv potřebujeme. Nevýhodou však je, že

dataset obsahuje pouze metadata publikací. Například podrobnější informace o autorech, jako je identifikátor, dosažená ocenění nebo příslušnost k univerzitě, v něm nenajdeme. Dále si dataset musíme sami pravidelně obnovovat, abychom vždy pracovali s aktuálními daty. Taktéž si musíme sami implementovat všechny operace nad daty, jako například vyhledávání nebo seskupování publikací do proudů publikací.

- Stažení stránek ve formátu HTML – Při použití tohoto přístupu se dostaneme ke všem datům, ke kterým se dostane běžný uživatel webové stránky. Data stahujeme ve formě HTML souborů. Ty ale často obsahují i množství nerelevantních informací, což může komplikovat extrahování námi požadovaných dat.

- Stažení stránek ve formátu XML – Další možností je stránky stahovat ve formátu XML, který neobsahuje tolik nepodstatných informací. Zde je největší nevýhodou, že DBLP podporuje tuto možnost pouze pro některé typy stránek, konkrétně tedy jen pro stránky proudů publikací, svazků, publikací a autorů.

- Využití API pro vyhledávání – Toto řešení je ideální pro implementaci vyhledávání v databázi. Zásadním omezením však může být nemožnost získat více jak prvních 10000 vyhovujících záznamů.

Dalším omezením, na které je potřeba si dávat pozor především u posledních tří zmíněných možností, je omezení na počet požadavků, které lze za určitý časový interval poslat. V případě vysokého vytížení služby může DBLP vracet prázdnou odpověď se stavovým HTTP kódem 429. Z tohoto důvodu autoři DBLP doporučují čekat mezi dvěma po sobě následujícími požadavky alespoň jednu nebo dvě sekundy [12].

3 Použité technologie

Nástroj pro analýzu DBLP databáze jsem implementoval jako webovou aplikaci pomocí frameworku Next.js verze 14 [13] a řady dalších technologií. Jedním z hlavních faktorů, podle kterých jsem se při volbě použitých technologií rozhodoval, byla jejich popularita. Tu jsem alespoň orientačně určoval podle počtu stažení odpovídajících balíčků. Celou aplikaci jsem napsal v editoru Visual Studio Code a testoval ji ve webových prohlížečích:

- Google Chrome,
- Firefox,
- Safari.

3.1 Framework Next.js

Next.js je framework postavený na knihovně React [14], který zjednodušuje vývoj webových aplikací. Umožňuje pomocí jedné technologie vyvíjet jak serverovou, tak i klientskou část aplikace. Základem jsou React komponenty, které slouží ke tvorbě uživatelského rozhraní. Next.js tyto komponenty doplňuje o další funkcionality a optimalizace. Pro vykonávání kódu na straně serveru používá Next.js běhové prostředí Node.js. Funkcionality a vlastnosti, kvůli kterým jsem se rozhodl pro tento framework, jsou:

- Sever-side rendering – React komponenty je možné renderovat zcela, nebo jen částečně na straně serveru, a to buď staticky při sestavování aplikace, anebo dynamicky za běhu. Díky tomuto přístupu uživatel vždy vidí aspoň část obsahu stránky ještě před tím, než se vykoná jakýkoliv JavaScript kód v prohlížeči, což vede ke příjemnějšímu uživatelskému zážitku.
- App Router – Systém řešící mapování URL adres na jednotlivé stránky aplikace. Toto mapování je určené pouze adresářovou strukturou, kde každý adresář reprezentuje jeden segment cesty URL adresy.
- Rozšířené `fetch` API – Next.js automaticky ukládá do mezipaměti (cache) odpovědi na požadavky, které jsou posílány pomocí `fetch` API na straně serveru. V případě opakovaného zasílání požadavku na stejnou adresu vrací data uložená v mezipaměti. Tím se snižuje množství zasílaných požadavků na další služby.
- TypeScript – Next.js má vestavěnou podporu jazyka TypeScript [15], který je rozšířením jazyka JavaScript o statickou kontrolu datových typů. Statická kontrola datových typů mi umožnila odhalit množství chyb ještě při psaní kódu a minimalizovat tak problémy za běhu aplikace. Zároveň považuji za výhodu, že jsem celou aplikaci mohl napsat v jednom jediném programovacím jazyce.

- Modularita – Koncept React komponent umožňuje rozdělit aplikaci na menší, nezávislé a znovupoužitelné části s jasně definovaným účelem. Komponenty lze definovat buď jako funkce, nebo jako třídy. Já jsem zvolil definici pomocí funkcí, protože mi připadne jednodušší a přehlednější. Pro sdílení funkcionality mezi komponentami jsou v tomto případě k dispozici tzv. React hooks. Opět jde o funkce, které pouze zapouzdřují požadovanou logiku a které voláme v komponentách. React poskytuje sadu základních React hooks (například `useState()` pro uchování stavu komponenty). Můžeme si však definovat i vlastní nebo využít balíčků třetích stran s již předdefinovanými React hooks. Já se rozhodl použít balíček usehooks-ts [16], který poskytuje mnoho React hooks pro jednodušší práci například s událostmi nebo úložištěm prohlížeče.

Pokud funkce nepracuje s API specifickým pouze pro prohlížeč nebo naopak s API specifickým pouze pro Node.js, tak ji můžeme volat na straně klienta i na straně serveru. Když potřebuji, aby taková funkce mohla být zavolána pouze na straně serveru, tak používám balíček server-only [17]. Balíček zajistí, že obsah souboru, do kterého je tento balíček naimportovaný, můžeme použít pouze na straně serveru.

3.2 Databázový systém MongoDB

Na straně serveru jsem se rozhodl implementovat vlastní řešení mezipaměti postavené na databázovém systému MongoDB [18]. Do této paměti ukládám stažené záznamy z DBLP. Mezipaměť poskytovaná frameworkem Next.js totiž umožňuje uložení pouze záznamů, jejichž velikost nepřesahuje 2 MB.

MongoDB je NoSQL databázový systém, který ukládá data ve formě dokumentů formátu BSON (formát podobný JSON). Umožňuje mít flexibilní schémata, která je možné jednoduše měnit a integrovat s aplikacemi.

Pro přístup k MongoDB z aplikace jsem zvolil knihovnu Mongoose [19] určenou pro použití s Node.js. V jazyce JavaScript, respektive TypeScript, lze díky této knihovně na databázi nejen volat dotazy pro manipulaci s daty, ale i definovat databázová schémata. Integrovat MongoDB do Next.js aplikace tak pro mě bylo velmi snadné.

3.3 Knihovna SWR

Pro načítání dat ze serveru nebo přímo z DBLP na straně klienta jsem se rozhodl použít knihovnu SWR [20]. SWR poskytuje sadu předpřipravených React hooks, které mimo jiné řeší automatické ukládání dat do mezipaměti a zpracování stavů HTTP požadavků. Výrazně tak zjednodušuje logiku načítání dat a díky mezipaměti snižuje množství posílaných požadavků.

3.4 Knihovna Cheerio

Extrahování dat z dokumentů ve formátu HTML nebo XML na straně serveru řeším pomocí knihovny Cheerio [21]. Tato knihovna převádí HTML nebo XML dokumenty na jednoduchý objektový model (DOM) a poskytuje rozhraní silně inspirované knihovnou jQuery pro jeho procházení a manipulaci. Právě zmíněná jednoduchost objektového modelu zajišťuje velice rychlou a efektivní práci s dokumenty.

3.5 Knihovna he

V odpovědích z DBLP jsou některé znaky vyjádřeny jako HTML entity. Pro dekódování těchto entit na straně serveru používám knihovnu he [22]. React totiž v rámci ochrany proti XSS útokům automaticky kóduje veškerý text, který se vkládá do komponent. Kdybych tedy předem nedekódoval vkládaný text s entitou, tak by se nevykreslily požadované znaky, ale kódy jejich entit.

3.6 Knihovna uuid

V aplikaci si uživatel může vytvářet skupiny autorů, které se ukládají do úložiště prohlížeče. Aby bylo možné tyto skupiny jednoznačně identifikovat, tak jim přiřazuji identifikátory typu UUID. Pro generování těchto identifikátorů používám knihovnu uuid [23].

3.7 Knihovna D3.js

Tvorbu grafů jsem si značně zjednodušil použitím knihovny D3.js [24]. Tato knihovna vůbec nepracuje s pojmem „graf“. Pouze poskytuje sadu nástrojů, které při vizualizaci skládáme dohromady. To činí tuto knihovnu vysoce univerzální a je tak pomocí ní možné tvořit vizualizace dat všemožných druhů. D3.js poskytuje nástroje pro:

- výběr a tvorbu DOM elementů a aplikování různých operací na ně,
- přidávání různých metod interakce s DOM elementy,
- tvorbu animovaných přechodů mezi různými stavy vizualizace,
- mapování dat na různé typy vizuálních hodnot, jako je například pozice, velikost nebo barva,
- mapování dat na různé typy měřítek a os,
- filtraci, seskupování a transformaci dat.

3.8 Knihovna Tailwind CSS

Vizualizaci komponent uživatelského rozhraní řeším primárně pomocí knihovny Tailwind CSS [25]. Tato knihovna poskytuje tzv. utility třídy, tedy jednoúčelové třídy, jejichž odpovídající CSS pravidla typicky nastavují hodnotu pouze jedné CSS vlastnosti. Tailwind CSS nabízí širokou škálu utility tříd pro různé vlastnosti, jako jsou barvy, typografie nebo layout. Díky tomuto přístupu jsem napsal minimum CSS kódu a nemusel se soustředit na jeho strukturu a ani vhodné pojmenovávání tříd. Přitom všem jsem navíc měl plnou kontrolu nad tím, jak jsou komponenty aplikace vizualizované. Na druhou stranu je nevýhodou zhoršená čitelnost kódu komponent, které vyžadují aplikaci většího množství tříd.

Next.js umožňuje nainstalovat a nakonfigurovat Tailwind CSS již při zakládání projektu pomocí nástrojů příkazové řádky. V rámci tohoto procesu taktéž automaticky instaluje knihovnu PostCSS [26], kterou využívá pro kompilaci CSS, a její plugin Autoprefixer [27], který doplňuje CSS pravidla o prefixy prohlížečů, aby byla zajištěna podpora nejnovějších funkcionalit.

V souvislosti s integrací Tailwind CSS do Next.js aplikace ještě používám následující knihovny:

- tailwind-merge – poskytuje JavaScript funkci, která slučuje řetězce s názvy Tailwind CSS tříd a řeší při tom možné konflikty [28],
- clsx – poskytuje JavaScript funkci, která umožňuje podmíněně kombinovat názvy tříd do jednoho řetězce [29],
- Class Variance Authority – zjednodušuje aplikaci různých tříd podle konkrétních variant komponenty [30].

3.9 Knihovna React Icons

Pro všechny ikony v aplikaci používám knihovnu React Icons [31], která nabízí širokou škálu ikon z mnoha balíčků, jako je například Font Awesome, Material Design icons nebo Ionicons. Já jsem vybíral ikony z balíčku Material Design icons, který je zveřejněný pod licencí Apache License 2.0.

3.10 Knihovna Jest

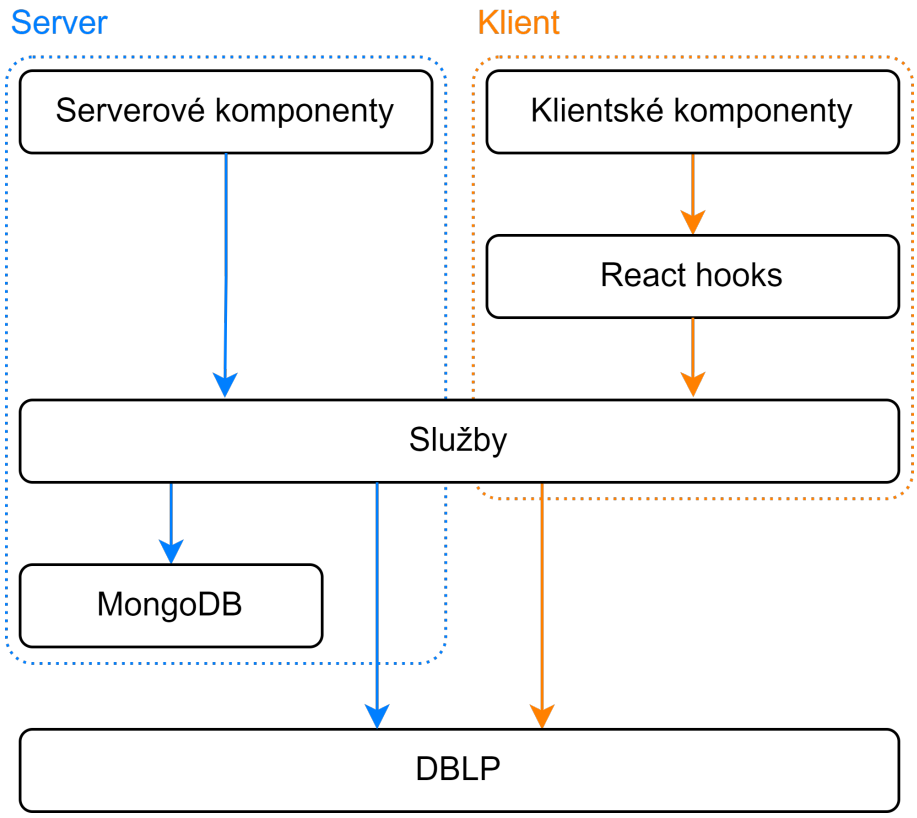
Automatizované jednotkové testy jsem implementoval pomocí knihovny Jest [32]. Tato knihovna je dobře zdokumentovaná, nevyžaduje složitou konfiguraci a umožňuje i testování React komponent.

4 Architektura aplikace

Z pohledu architektury, která dělí aplikaci na prezentační, aplikační (logickou) a datovou vrstvu, je prezentační vrstva aplikace tou nejvýznamnější a nejobsáhlejší. Největší část kódu tak tvoří React komponenty. Tomu jsem přizpůsobil i celkovou architekturu aplikace. Zmíněným vrstvám odpovídají:

- React komponenty – prezentační vrstva,
- React hooks a služby (services) – aplikační vrstva,
- MongoDB a DBLP – datová vrstva.

Klientská část aplikace neslouží čistě jen pro prezentaci informací získaných ze serveru, ale obsahuje i část aplikační vrstvy. Obsahuje například logiku pro vyhledávání, filtrování, interaktivní grafy nebo vytváření skupin autorů. Co největší část aplikační vrstvy jsem se však snažil soustředit na server. Celkovou architekturu aplikace můžeme vidět znázorněnu na obrázku 6.



Obrázek 6: Architektura aplikace a znázornění komunikace mezi jejími částmi

4.1 Datová vrstva

DBLP databáze představuje v této aplikaci primární zdroj dat. Poskytuje informace o prezentovaných autorech a proudech publikací. Na tuto databázi se posílají dotazy primárně ze serverové části aplikace, aby mohly být získané záznamy ukládány do mezipaměti. Vlastní řešení mezipaměti jsem implementoval pomocí databázového systému MongoDB. K MongoDB přistupuji skrze modely definované pomocí knihovny Mongoose.

Na straně klienta se posílají dotazy na DBLP databázi pouze pro implementaci rychlého vyhledávání, a to pomocí `fetch` API prohlížeče. Konkrétně zde využívám vyhledávacího API, které DBLP poskytuje. Výsledky vyhledávání z tohoto API aplikace přijímá ve formátu JSON.

4.2 Prezentační vrstva

Prezentační vrstvu aplikace tvoří React komponenty, které jsou definované jako funkce. V Next.js se ve výchozím stavu všechny komponenty renderují na serveru a klientovi se posílá již vyrenderovaný HTML kód. Tyto komponenty se označují jako *serverové komponenty*. Příklad serverové komponenty můžeme vidět na ukázce kódu 1.

```
1  export default async function SearchAuthorPage(
2     { searchParams }: SearchAuthorPageParams
3  ) {
4     const params = searchToItemsParams(
5        searchParams,
6        DEFAULT_ITEMS_COUNT_PER_PAGE);
7     const result = await getSearchResult(params);
8
9     return (
10       <SearchResultList
11          result={result}
12          searchParams={searchParams}
13          paginationUrl={SEARCH_AUTHOR} />
14    )
15 }
```

Zdrojový kód 1: Serverová komponenta stránky, která na základě parametrů URL adresy načte výsledky vyhledávání autorů a předá je další komponentě pro jejich zobrazení.

Serverové komponenty však nemohou být interaktivní a volat JavaScript na straně klienta. K tomu slouží *klientské komponenty*, které se renderují až na klientovi. Tyto komponenty definujeme v souborech anotovaných řetězcem `use client`. Všechny komponenty, které jsou potomky klientských komponent, jsou opět klientské komponenty.

V aplikaci používám serverové komponenty v co nejvyšší míře. Tím, že se komponenty renderují již na serveru, se snižuje množství poslaného JavaScript kódu a kódu, který se musí vykonat na klientovi. Zároveň, pokud je to možné, řeším načítání a zpracování dat již v serverových komponentách, abych snížil množství vykonaných HTTP požadavků. Takto načtená data lze předat do klientské komponenty jako běžnou vlastnost React komponenty. Next.js vyřeší přenos dat ze serveru do komponenty automaticky. Klientské komponenty používám kdykoliv, když potřebuji, aby byly schopné reagovat na vstup uživatele nebo pracovat s API prohlížeče.

4.3 Aplikační vrstva

Aplikační vrstvu tvoří primárně služby (services) a React hooks. Jako *služby* označuji sady funkcí, které poskytují nějakou ucelenou funkcionalitu, typicky zpracování dat. Příkladem může být sada funkcí, která řeší načítání metadat autorů z DBLP a jejich následné zpracování pro další použití v aplikaci.

React hooks implementují logiku aplikace na straně klienta. React komponenty jejich prostřednictvím přistupují k funkcionalitám služeb. React hooks buď volají služby přímo (například služba pro generování grafu spoluautorství), anebo skrze HTTP koncový bod, pokud daná služba musí být vykonána na straně serveru (například stažení metadat autora). Ve druhém případě využívám `fetch` API a knihovnu SWR. Příkladem takového React hook je funkce `useAuthor`, jejíž definici můžeme vidět na ukázce kódu 2.

```
1  export default function useAuthor(authorId: string) {
2     const { data, error, isLoading } = useSWRImmutable(
3        authorId,
4        authorFetcher);
5
6     return {
7        author: data,
8        isLoading,
9        error
10    };
11 }
12
13 async function authorFetcher(authorId: string) {
14    await waitForNextFetchClient();
15    const response = await fetch("/api/author/" + authorId);
16    return await response.json() as DblpAuthor;
17 }
```

Zdrojový kód 2: React hook `useAuthor`, který umožňuje prostřednictvím knihovny SWR stáhnout metadata autora ze serveru.

Dále React hooks používám čistě jen pro oddělení logiky React komponenty od její vizualizace. Výsledkem je lepší znovupoužitelnost a čitelnost kódu.

4.4 Organizace kódu

Kód aplikace je rozdělen do funkcí, které jsou dále podle poskytované funkcionality rozděleny do souborů. Pokud je funkce z jednoho souboru potřeba v jiném souboru, je vyexportovaná do JavaScript modulu. Soubory jsou nakonec rozdělené podle účelu do adresářů. Všechen kód aplikace nalezneme v adresáři `src/`, který obsahuje konfigurační soubory použitých knihoven a následující adresáře:

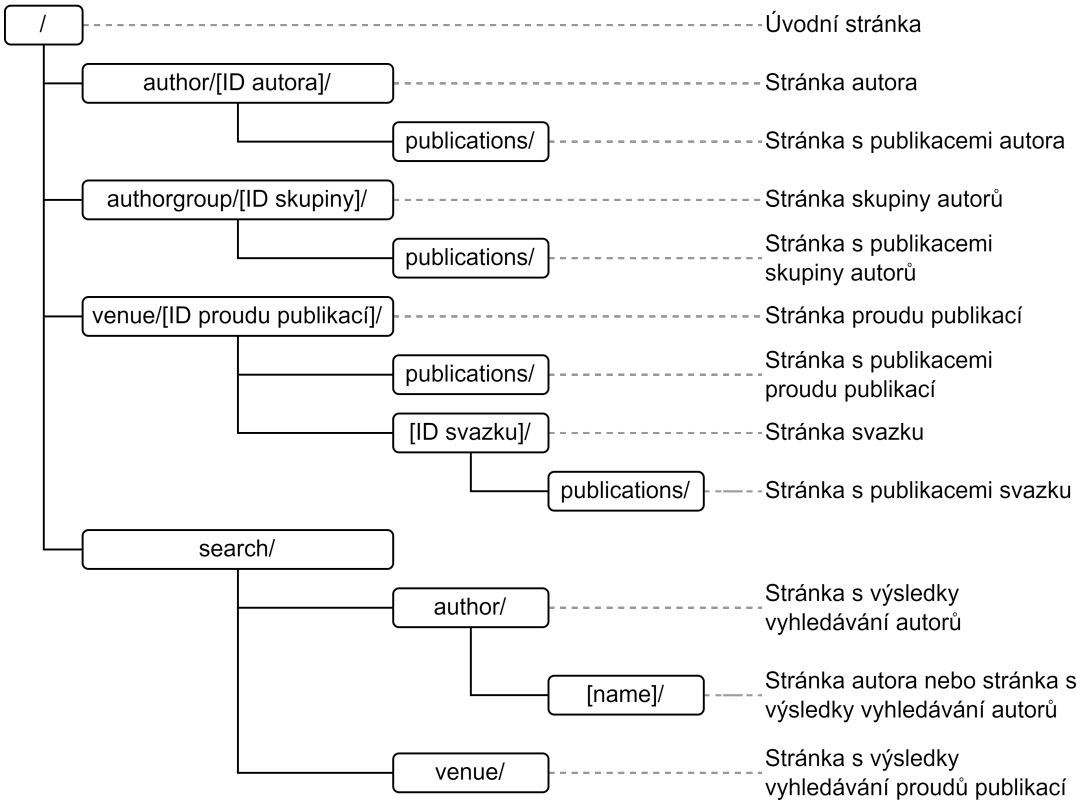
- `__tests__/` – Adresář s jednotkovými testy napsanými pomocí knihovny Jest. Pokrytí kódu automatizovanými testy je jedním z nedostatků aplikace. Jednotkovými testy jsou zatím pokryty pouze utility funkce.
- `app/` – Stěžejní adresář Next.js aplikace, která využívá App Router. Obsah tohoto adresáře popisuje mapování segmentů URL adres na stránky aplikace. Ty jsou tvořeny React komponentami umístěnými v souborech s názvem page.tsx.
- `components/` – Adresář se všemi React komponentami, které jsou sdílené mezi více stránkami.
- `constants/` – Adresář s konstantami. Některé konstanty, které se používají pouze v rámci jednoho souboru, zde nemusí být zahrnuty.
- `css/` – Adresář s CSS soubory, ve kterých najdeme například definované barevné téma aplikace nebo vizualizaci tlačítek.
- `db/` – Adresář s definicí schémat MongoDB databáze a modelů, pomocí kterých lze přistupovat k datům databáze. V souboru `mongodb.ts` se nachází stěžejní funkce `connectDb` pro vytvoření spojení s databází. Je nutné ji volat před přístupem k databázi.
- `dtos/` – Adresář s TypeScript typy pro objekty, které se používají k výměně dat mezi různými částmi aplikace.
- `enums/` – Adresář s výčtovými typy.
- `hooks/` – Adresář s React hooks. Některé React hooks, které se používají pouze v rámci jednoho souboru (respektive React komponenty), zde nemusí být zahrnuty.
- `public/` – Adresář se statickým obsahem, který je přístupný z klienta.
- `services/` – Adresář se všemi službami aplikace.
- `utils/` – Adresář s *utility funkcemi*, tedy jednoúčelovými pomocnými funkcemi. Najdeme zde například pomocné funkce pro práci s řetězci, poli nebo URL adresami.

4.5 Stránky a API aplikace

Na obrázku 7 můžeme vidět znázorněnu organizaci všech stránek aplikace z pohledu jejich URL adres. Aplikace nabízí následující stránky:

- Úvodní stránka – Obsahuje jednoduchý rozcestník na stránky indexů autorů a proudů publikací.
- Stránka autora – Obsahuje vizualizace různých statistik autora a interaktivní graf jeho spoluautorů.
- Stránka s publikacemi autora – Obsahuje přehled publikací autora s možností jejich filtrování.
- Stránka skupiny autorů – Obsahuje vizualizace různých statistik skupiny autorů, kterou si uživatel vytvořil, a interaktivní graf jejich spoluautorů.
- Stránka s publikacemi skupiny autorů – Obsahuje přehled publikací skupiny autorů s možností jejich filtrování.
- Stránka proudu publikací – Obsahuje vizualizace různých statistik proudu publikací a přehled případných svazků daného proudu publikací. V případě existence svazků umožňuje jejich výběr a zobrazení vizualizací různých jejich statistik a interaktivního grafu autorů publikací z vybraných svazků.
- Stránka s publikacemi proudu publikací – Obsahuje přehled publikací proudu publikací s možností jejich filtrování. V případě existence svazků umožňuje jejich výběr a zobrazení přehledu jejich publikací.
- Stránka svazku – Obsahuje vizualizace různých statistik svazku a interaktivní graf autorů jeho publikací.
- Stránka s publikacemi svazku – Obsahuje přehled publikací svazku s možností jejich filtrování.
- Stránka s indexem autorů nebo výsledky jejich vyhledávání.
- Stránka s indexem proudů publikací nebo výsledky jejich vyhledávání.

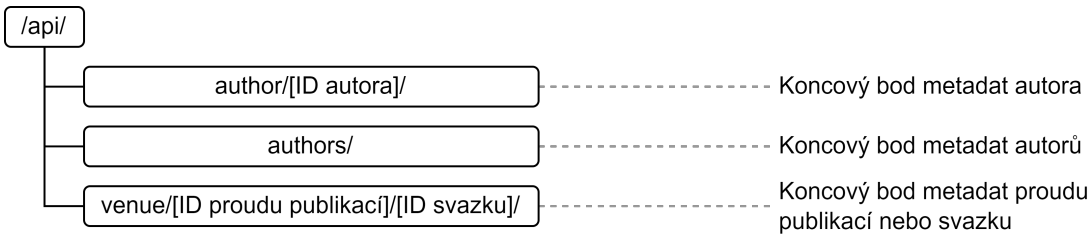
Aplikace taktéž nabízí koncový bod API, který uživatele podle zadaného jména přesměruje buď na stránku autora tohoto jména, pokud v DBLP databázi existuje pouze jeden autor daného jména, nebo na stránku s výsledky vyhledávání autorů daného jména. Tento koncový bod se využívá v situacích, kdy není k dispozici unikátní identifikátor autora, ale jen jeho jméno.



Obrázek 7: Organizace stránek aplikace z pohledu jejich URL adres

Dále API aplikace poskytuje ještě tři koncové body, jejichž URL adresy můžeme vidět na obrázku 8:

- koncový bod vracející metadata autora s předaným identifikátorem,
- koncový bod vracející metadata autorů s předanými identifikátory,
- koncový bod vracející metadata proudu publikací nebo svazku s předaným identifikátorem.



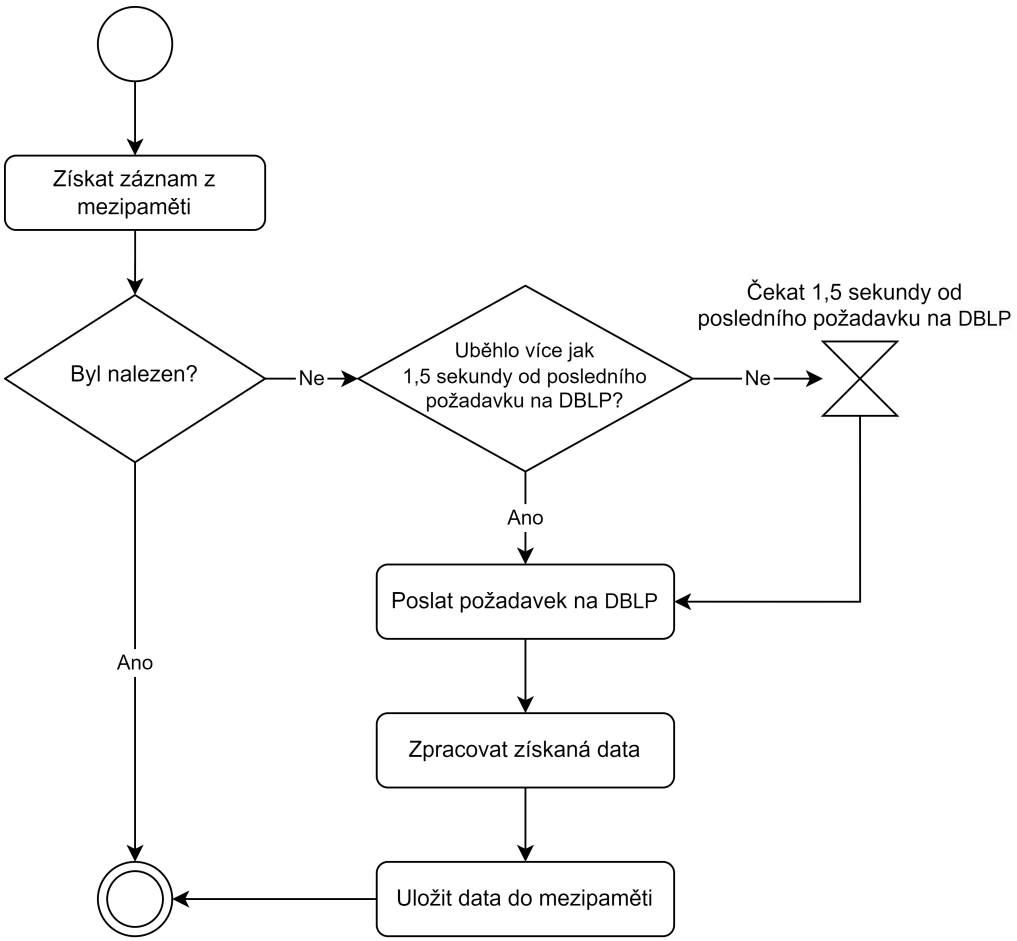
Obrázek 8: URL adresy koncových bodů API aplikace

5 Práce s daty

Z možností stažení dat z DBLP, jejichž přehled najdeme v kapitole 2.6, jsem se rozhodl využít následující:

- stažení stránek ve formátu HTML – pro získání indexů autorů a proudů publikací,
- stažení stránek ve formátu XML – pro získání metadat autorů, proudů publikací a svazků,
- API pro vyhledávání – pro implementaci vyhledávání autorů a proudů publikací.

Jednodušší implementace těchto možností podle mého názoru převažuje výhody použití celého datasetu v podobě souboru `dblp.xml`. Především nemusím řešit pravidelné obnovování datasetu a uživatel tak má vždy zajištěn přístup k aktuálním datům. Taktéž nemusím řešit vlastní implementaci vyhledávání.



Obrázek 9: Diagram aktivit popisující načítání metadat autora, proudu publikací nebo svazku

Aby nedocházelo k přetížení služby, doporučují autoři DBLP čekat mezi dvěma po sobě následujícími požadavky alespoň jednu nebo dvě sekundy. Já se v situacích, kdy by potenciálně mohlo docházet k odeslání většího množství požadavků v kratším časovém intervalu, rozhodl čekat mezi požadavky alespoň 1,5 sekundy. Toto čekání zajišťuje asynchronní funkce `waitForNextFetch`, která se volá před odesláním požadavku na DBLP. Funkce si do databáze ukládá čas poslání posledního požadavku na DBLP. Pokud při volání této funkce uběhlo méně než 1,5 sekundy od poslání posledního požadavku na DBLP, funkce čeká dostatečně dlouhou dobu, aby tomu tak bylo.

Problém přetížení DBLP zasláním většího množství požadavků dále řeším využitím mezipaměti. Využívám jak zcela automatické řešení v Next.js a knihovně SWR, tak i vlastní řešení postavené na databázovém systému MongoDB. Vlastní řešení mezipaměti používám pouze u záznamů získaných ze stránek ve formátu XML. Tyto stránky totiž někdy přesahují velikost 2 MB a mezipaměť poskytovaná frameworkem Next.js neumožňuje uložení záznamů, jejichž velikost přesahuje 2 MB. Popis procesu získání metadat ze stránek ve formátu XML můžeme vidět na obrázku 9.

5.1 Vlastní identifikátory

V rámci aplikace jsem se rozhodl používat vlastní identifikátory autorů, proudů publikací a svazků. Jejich identifikátory v DBLP totiž mohou obsahovat znak „/“. Protože však potřebuji identifikátory předávat přes URL adresu a znak „/“ v identifikátoru by tento proces komplikoval, nahrazuji v každém identifikátoru všechny znaky „/“ posloupností dvou znaků „_“. Například namísto identifikátoru „journals/aiopen“ tak pracuji s identifikátorem „journals__aiopen“.

5.2 Zpracování stažených dat

Extrahování dat ze souborů ve formátu HTML nebo XML na straně serveru řeším pomocí knihovny Cheerio. Funkce provádějící tuto extrakci se nacházejí v souborech:

- `src/services/authors/parsing.ts` – obsahuje funkce pro extrahování indexu autorů a metadat autorů,
- `src/services/venues/parsing.ts` – obsahuje funkce pro extrahování indexu proudů publikací a metadat proudů publikací a svazků,
- `src/services/publications/parsing.ts` – obsahuje funkce pro extrahování metadat publikací, které jsou sdílené předchozími soubory.

Nejkomplikovanější je extrahování metadat ze stránek proudů publikací ve formátu XML. Tyto stránky totiž nemají pevně danou strukturu a kombinují standardní HTML značky se značkami vlastními. V některých případech jsou

odkazy na svazky umístěné například v tabulce `<table>` a v jiných zase v seznamech `<ul>`, které mohou být do sebe i několikanásobně zanořené. Dále jsou v mnoha případech metadata publikací uvedena již na stránce proudu publikací. Takové publikace většinou samy reprezentují stránky svazků a slouží jako odkazy na ně. Taktéž se může jednat o proudy publikací, které se již dále nedělí na svazky.

Výstupem funkcí pro extrahování metadat jsou objekty sloužící k výměně dat mezi různými částmi aplikace. Jejich TypeScript typy a funkce pro jejich vytvoření se nachází v adresáři `src/dtos/`. Stěžejní typy jsou následující:

- `DblpPublication` – pro metadata publikace,
- `DblpAuthor` – pro metadata autora,
- `DblpVenue` – pro metadata proudu publikací,
- `DblpVenueVolume` – pro metadata svazku,
- `SimpleSearchResult` – pro indexy a výsledky vyhledávání autorů nebo proudů publikací.

5.3 Data uživatele

Data uživatele se ukládají do úložiště prohlížeče `localStorage`. Konkrétně se jedná o následující data:

- naposledy navštívení autoři,
- naposledy navštívené proudy publikací nebo svazky,
- uložení autoři pro snazší přístup,
- uložené proudy publikací nebo svazky pro snazší přístup,
- vytvořené skupiny autorů.

Nevýhodou tohoto řešení je, že uživatel nemá takto uložená data snadno přístupná z různých prohlížečů. Proto jsem zvažoval i možnost uložení dat uživatele v databázi na straně serveru. Toto řešení by vyžadovalo implementaci autentizace a autorizace uživatele, aby měl přístup jen ke svým datům, což by však zkomplikovalo návrh celé aplikace. Nutnost registrace by navíc mohla odradit některé uživatele od používání aplikace. Nakonec jsem se tak rozhodl ukládat data do úložiště prohlížeče a přidal do aplikace možnost exportu a importu dat uživatele.

Do `localStorage` se z důvodu omezené velikosti neukládají celé objekty autorů, proudů publikací a svazků, ale jen jich identifikátory a názvy. Zbylá metadata se stahují voláním koncových bodů API aplikace.

6 Klíčové komponenty aplikace

Aplikace obsahuje řadu základních univerzálních komponent. Jedná se například o různá tlačítka, kontejnery, záložky, tabulky nebo dialogy. Jejich kombinací vznikají komponenty složitější. Z nich jsou klíčové především komponenty pro vizualizaci statistik a komponenta grafu spoluautorství. Aplikace nepoužívá kromě Next.js žádnou další knihovnu poskytující již hotové React komponenty.

6.1 Vizualizace statistik

Většina statistik, které aplikace umožňuje vizualizovat, jsou kolekce párů typu „klíč–číselná hodnota“. Například statistika udávající počty zveřejněných publikací v jednotlivých letech. Aplikace nabízí celkem tři React komponenty pro vizualizaci takových kolekcí:

- `LineChart` – liniový graf,
- `BarChart` – sloupcový graf,
- `PieChart` – koláčový graf.

Komponenty na vstup dostávají seznam položek, které seskupí podle zkoumané vlastnosti a každé skupině přiřadí číselnou hodnotu. Tím vzniká kolekce párů typu „klíč–číselná hodnota“, kde klíčem je hodnota zkoumané vlastnosti. Tuto kolekci komponenta vizualizuje formou požadovaného grafu.

Vstupní data, která chceme vizualizovat, jsou komponentám předávána jako objekt typu `ChartData`. Každá komponenta si tento typ rozšiřuje o vlastní vlastnosti. Typ `ChartData` definuje především vlastnosti pro:

- seznam zkoumaných položek (například seznam publikací autora),
- funkci, která pro předanou položku vrací hodnotu zkoumané vlastnosti,
- funkci, která předané skupině přiřazuje číselnou hodnotu (pokud není definovaná, volí se počet položek skupiny).

Samotné grafy tvoří kombinace SVG a HTML elementů. Mapování dat na různé typy vizuálních hodnot, měřítek nebo os zajišťuje knihovna D3.js. Nevyužívám však možnosti manipulace s DOM, kterou tato knihovna poskytuje. Manipulace s DOM je zcela v režii knihovny React.

Grafy a tabulky prezentující konkrétní statistiku se vždy sdružují do jedné komponenty, která umožňuje mezi různými prezentacemi přepínat. Příkladem může být komponenta `PublicationsOverTimeStats` pro statistiku udávající počty zveřejněných publikací v jednotlivých letech. Komponenty všech statistik najdeme v adresáři `components/data-visualsation/stats/`.

6.2 Graf spoluautorství

O vytvoření datových struktur, které reprezentují graf spoluautorství množiny autorů, se stará funkce `convertToCoauthorsGraph`. Tato funkce přijímá pole publikací, jež převádí na pole uzlů reprezentujících autory a pole neorientovaných hran mezi těmito uzly. Hrana je vytvořena mezi každými dvěma autory stejné publikace ze vstupního pole. Dále funkce ještě vrací objekt slovníku. Tento slovník mapuje identifikátory autorů na jejich uzly a umožňuje tak rychlý přístup k uzlům na základě jejich identifikátorů.

Zobrazení interaktivní vizualizace grafu a souvisejících informací, jako je například seznam autorů v grafu nebo informace o aktuálně vybraném autorovi, zajišťuje React komponenta `CoauthorsGraphShell`. Stav grafu, se kterým tato komponenta pracuje, spravuje React hook `useCoauthorsGraph`. Samotný graf se vykresluje na HTML element `<canvas>`. Určení souřadnic uzlů grafu je založeno na simulaci fyzikálních sil implementované pomocí knihovny D3.js [33]. V simulaci se na uzly grafu aplikují celkem tři síly:

- `forceLink` – přitahuje k sobě uzly propojené hranou,
- `forceManyBody` s negativní intenzitou – vzájemně od sebe odpuzuje všechny uzly,
- `forceCenter` – všechny uzly přitahuje do specifikovaného bodu.

Jelikož je tato simulace výpočetně náročná, rozhodl jsem se ji vykonávat na pozadí v rámci pracovního vlákna poskytovaného funkcionalitou Web Worker [34]. Pracovní vlákno přerušuje to hlavní pouze zasíláním průběžného postupu simulace a konečného výsledku v podobě pole uzlů a hran s aktualizovanými souřadnicemi. Na ukázce kódu 3 můžeme vidět kód pro vytvoření simulace.

```
1 const simulation = d3.forceSimulation(event.data.nodes)
2    .force('link', d3.forceLink()
3       .id((d) => d.person.id)
4       .links(event.data.links))
5    .force('charge', d3.forceManyBody().strength(-50))
6    .force('center', d3.forceCenter(
7       event.data.graphWidth / 2,
8       event.data.graphHeight / 2))
9    .stop();
```

Zdrojový kód 3: Kód pro vytvoření simulace fyzikálních sil, která určí souřadnice uzlů grafu.

7 Zprovoznění aplikace

Aplikaci je možné zprovoznit dvěma způsoby, a to buď pomocí Docker, anebo bez něj. První zmiňovaný způsob je určen pro testování a nasazení produkční verze aplikace. Druhý způsob je pak určen především pro vývoj.

7.1 Za použití Docker

Ke zprovoznění aplikace pomocí Docker je potřeba mít nainstalovánu jeho aktuální verzi spolu s Docker Compose. V přiložených elektronických datech v adresáři `src/` jsou k dispozici dva Docker Compose soubory:

- `docker-compose-dev.yml` – pro lokální testování aplikace v produkčním režimu (aplikace běží na portu 3000),
- `docker-compose-prod.yml` – pro nasazení aplikace na produkci (aplikace běží na portu 80).

Oba soubory zajistí vytvoření a spuštění MongoDB a samotné aplikace. Potřebné proměnné prostředí a porty aplikací jsou předdefinované již v těchto souborech. Celý postup zprovoznění aplikace je pro oba soubory totožný:

1. Ujistíme se, že máme na našem zařízení spuštěný Docker.
2. V terminálu přejdeme do adresáře `src/` s výše zmíněnými soubory.
3. Sestavíme Docker kontejnery podle požadovaného souboru:

   ```
   docker compose -f ./docker-compose-dev.yml build
   ```

4. Spustíme Docker kontejnery požadovaného souboru:

   ```
   docker compose -f ./docker-compose-dev.yml up
   ```

7.2 Bez použití Docker

Pro účely vývoje aplikace je vhodnější ji spouštět bez použití Docker. Pokud totiž spustíme aplikaci ve vývojovém režimu, budeme moci využít funkcionalitu Fast Refresh. Ta zajistí, že se veškeré změny kódu ihned promítnou do běžící aplikace.

V tomto případě je nutné mít k dispozici běžící MongoDB. Já zvolil komunitní verzi, kterou je možné si nainstalovat lokálně. Postup instalace najdeme v dokumentaci [35]. Dále je nutné mít nainstalované Node.js spolu se správcem balíčků npm [36]. Celý postup zprovoznění aplikace je následující:

1. Ujistíme se, že máme nainstalované Node.js a npm a běžící MongoDB.

2. V adresáři `src/` vytvoříme soubor `.env` pro proměnné prostředí. Do tohoto souboru přidáme proměnnou `MONGODB_URI` s adresou databáze, například pro lokálně běžící MongoDB:

   ```
   MONGODB_URI=mongodb://127.0.0.1:27017/dblp-tool
   ```

3. V terminálu přejdeme do adresáře `src/`.

4. Nainstalujeme potřebné závislosti příkazem `npm install`.

5. Aplikaci spustíme buď ve vývojovém, anebo produkčním režimu:

   (a) Ve vývojovém režimu spustíme aplikaci příkazem:

   ```
   npm run dev
   ```

   (b) Pro spuštění aplikace v produkčním režimu ji musíme nejprve sestavit:

   ```
   npm run build
   ```

   Aplikaci následně spustíme příkazem:

   ```
   npm run start
   ```

6. Adresu, na které aplikace běží, vidíme vypsanou v terminálu.

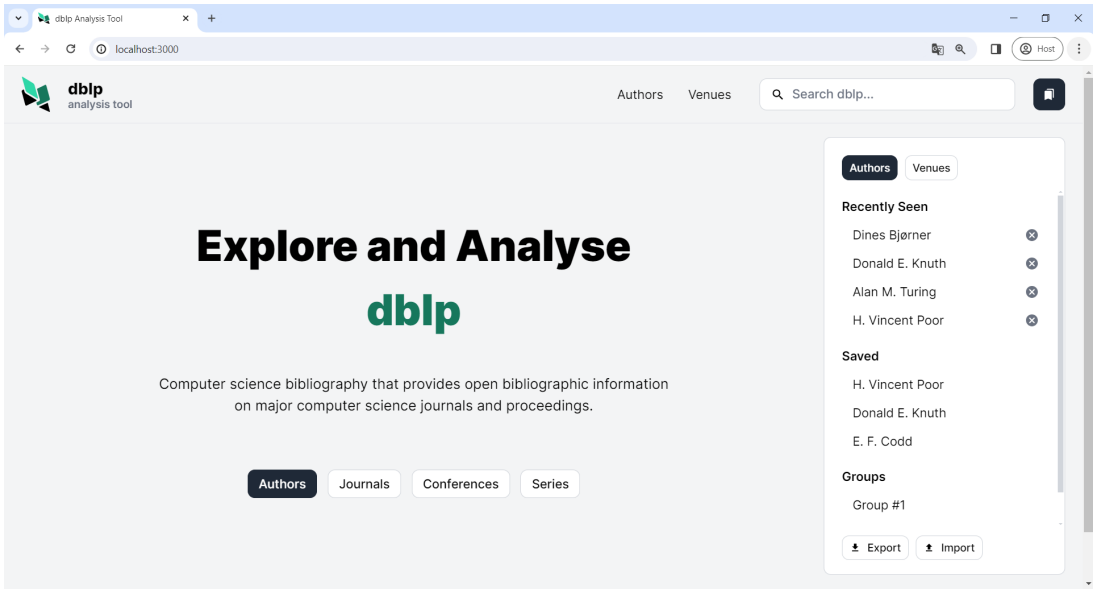
8 Uživatelská příručka

Aplikaci jsem vyvíjel s předpokladem, že ji uživatelé budou používat především v desktopových prohlížečích. V nich tak aplikace nabízí nejlepší uživatelský zážitek. Aplikace je sice responzivní, na mobilních zařízeních však její používání není ideální. Zejména grafy nejsou zcela přizpůsobeny pro zobrazení na zařízeních s úzkými displeji. K lepšímu uživatelskému zážitku na všech typech zařízení přispívá podpora tmavého režimu.

8.1 Úvodní stránka a postranní menu

Po zobrazení aplikace v prohlížeči nás přivítá úvodní stránka s jednoduchým rozcestníkem na stránky všech dostupných indexů (viz obrázek 10). Odkazy na indexy najdeme taktéž v hlavičce stránky, kde se dále nachází tlačítka pro zobrazení vyhledávacího dialogu a postranního menu. V postranním menu máme přístupné:

- nedávno navštívené autory,
- uložené autory,
- vytvořené skupiny autorů,
- nedávno navštívené proudy publikací,
- uložené proudy publikací.



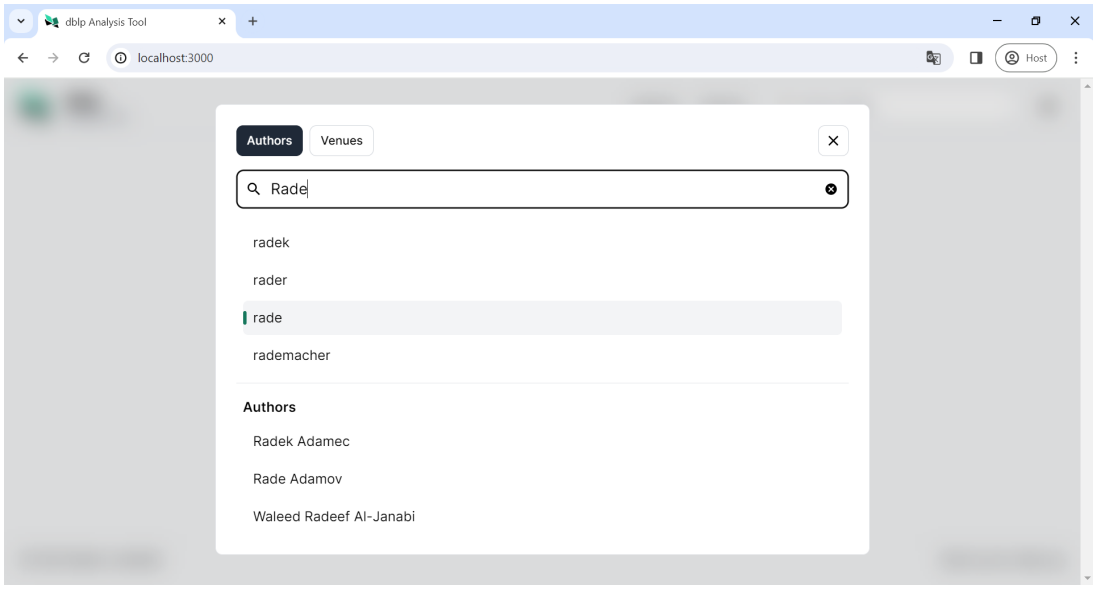
Obrázek 10: Úvodní stránka aplikace s otevřeným postranním menu

Uložené autory, skupiny autorů a proudy publikací si můžeme nechat vyexportovat nebo opětovně naimportovat odpovídajícími tlačítky ve spodní části menu. Před vykonáním exportu si lze zvolit, jaké konkrétní položky se mají exportovat. Obdobně i při importu. Položky se exportují do souboru ve formátu JSON.

8.2 Vyhledávání

Vyhledávač autorů a proudů publikací v této aplikaci funguje stejně jako ten v DBLP. Porovnává tedy každý hledaný výraz jako předponu potenciálního výsledku vyhledávání. Jestliže hledaný výraz obsahuje více slov oddělených prázdným znakem nebo znakem „+“, pak všechny musí být předponami v hledaném výsledku. Nerozlišují se velká a malá písmena a ani diakritika. Vyhledávání není fulltextové.

Vyhledávací dialog (viz obrázek 11) obsahuje především textové pole pro zadání hledaného výrazu s možností výběru, zda se mají vyhledávat autoři nebo proudy publikací. Po zadání výrazu se v dialogu ještě zobrazuje seznam slov, na která lze výraz doplnit, a seznam nejpravděpodobněji hledaných záznamů. Pokud takový záznam zvolíme, jsme ihned přesměrováni na odpovídající stránku autora, respektive proudu publikací. Jinak jsme při potvrzení hledaného výrazu přesměrováni na stránku se všemi výsledky hledání. Jestliže nezadáme žádný výraz, je vypsán odpovídající index. Stránka s výsledky vyhledávání proudů publikací je rozdělena na tři sekce podle typů proudů – časopisy, konference a série.



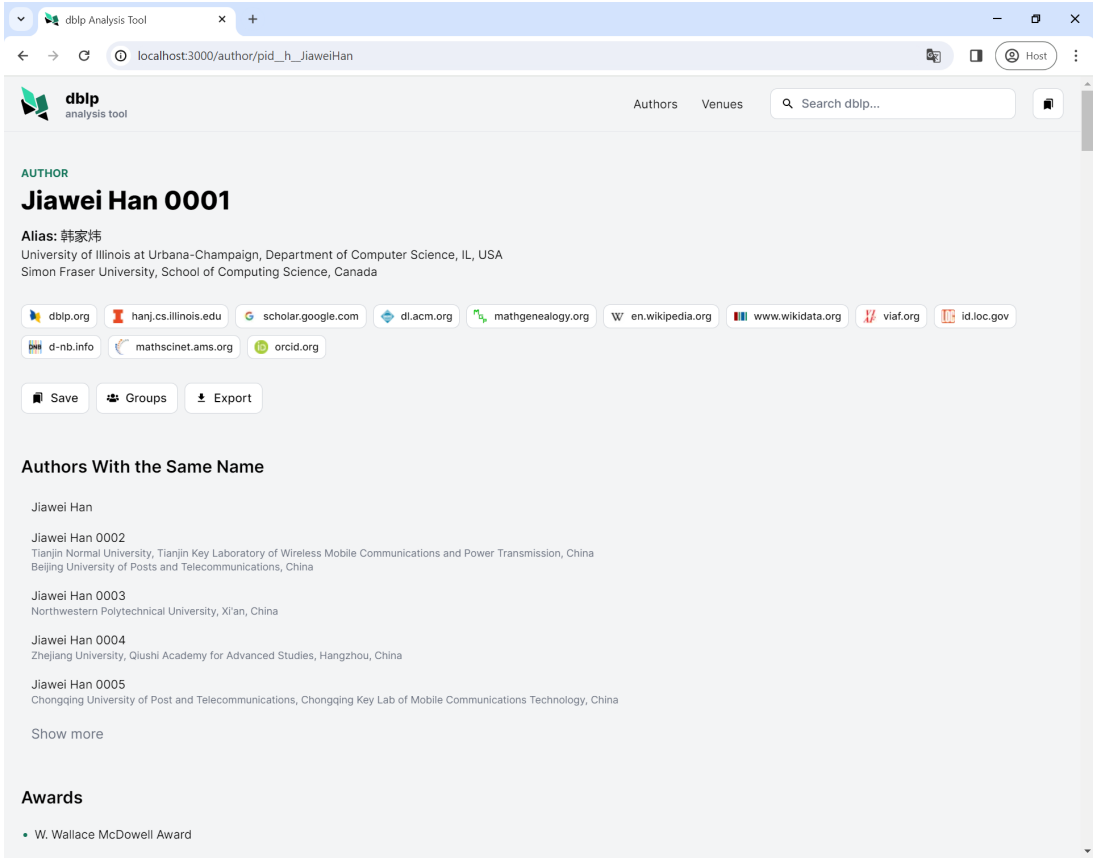
Obrázek 11: Vyhledávací dialog v aplikaci

8.3 Stránka autora

Stránka autora je rozdělena do několika sekcí. První je hlavička, ve které nalezneme zejména autorovo jméno, případný alias, příslušnost k univerzitě a odkazy na různé domovské stránky autora. Vždy je zde minimálně odkaz na stránku autora v DBLP. Dále jsou v hlavičce ještě tři tlačítka pro:

- uložení autora,
- zobrazení dialogu pro přidání autora do skupiny,
- export metadat autora do souboru ve formátu JSON.

Po hlavičce následuje sekce se seznamem autorů stejného jména a sekce se seznamem ocenění. Ne u všech autorů jsou tyto sekce k dispozici. Příklad úvodní části stránky můžeme vidět na obrázku 12.

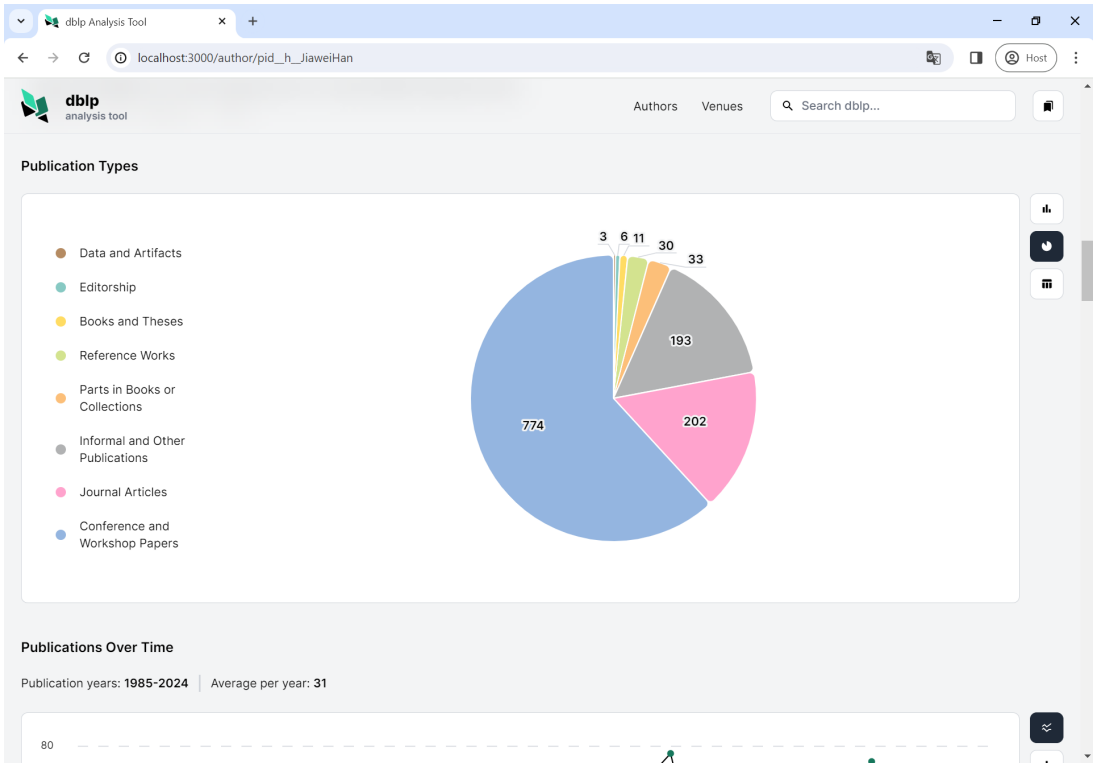


Obrázek 12: Stránka autora Jiawei Han 0001 v aplikaci

Další sekcí je sekce s různými statistikami publikací. Titulek sekce je zároveň odkazem na stránku se všemi publikacemi autora. V této sekci nalezneme vypsány nejvýše tři nejnovější publikace autora a tři různé statistiky publikací:

- počty publikací jednotlivých typů (viz obrázek 13),
- počty publikací publikovaných v jednotlivých letech,
- přehled proudů publikací, do kterých jsou publikace zařazeny.

Všechny statistiky jsou prezentovány tabulkou a alespoň jedním typem grafu. Pokud ve sloupcovém grafu klikneme na některý sloupec, jsme přesměrováni na stránku s vyfiltrovanými publikacemi, které danému sloupci odpovídají. Obdobně funguje i koláčový graf.



Obrázek 13: Komponenta koláčového grafu, který vizualizuje počty publikací autora jednotlivých typů.

Poslední sekcí je sekce s grafem spoluautorství a tabulkou všech spoluautorů. Graf spoluautorství je popsán v kapitole 8.7. V tabulce spoluautorů je u každého autora uveden počet společných publikací se zobrazeným autorem a počet společných spoluautorů těchto publikací. Nemusí se však nutně jednat o celkový počet společných spoluautorů. Dva různí autoři totiž mohou mít společného spoluautora a přitom všichni tři nemusí být autory stejné publikace.

8.4 Stránka skupiny autorů

Stránka skupiny autorů je stejně jako stránka autora rozdělena do několika sekcí. V hlavičce nalezneme název skupiny a tlačítka pro:

- přejmenování skupiny,
- smazání skupiny,
- export metadat členů skupiny do souboru ve formátu JSON.

Za hlavičkou následuje přehled všech členů skupiny, ze kterého si můžeme vybrat konkrétní členy, jejichž metadata mají být prezentována. Rychlost načtení metadat členů závisí na tom, zda jsou zrovna uložena v mezipaměti. Pokud nejsou, tak je rychlost významně omezena uměle zavedeným čekáním mezi poslanými požadavky na DBLP.

Další sekcí je sekce se statistikami publikací vybraných členů skupiny. Titulek sekce je zároveň odkazem na stránku se všemi publikacemi skupiny. Sekce obsahuje stejné statistiky jako na stránce autora a navíc ještě statistiku počtů publikací jednotlivých členů. Publikace je možné v této statistice filtrovat. Poslední sekce obsahuje graf spoluautorství (viz kapitola 8.7) a tabulku všech spoluautorů vybraných členů. V tabulce spoluautorů je u každého autora uveden celkový počet publikací, které jsou společné s alespoň jedním z vybraných členů skupiny, a počet společných spoluautorů těchto publikací.

8.5 Stránka proudu publikací

V hlavičce stránky proudu publikací nalezneme název proudu, odkaz na stránku proudu v DBLP a tlačítka pro:

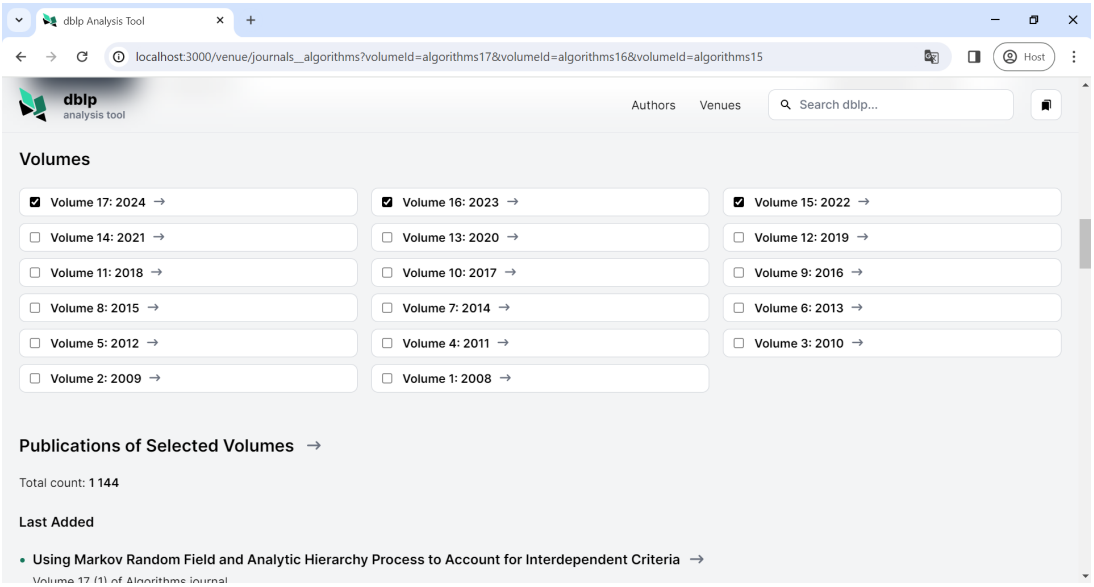
- uložení proudu publikací,
- export metadat proudu publikací a případných svazků do souboru ve formátu JSON.

Po hlavičce následuje sekce se statistikou počtu publikací publikovaných v jednotlivých letech v daném proudu publikací a statistikou nejaktivnějších autorů v daném proudu publikací. Další obsah stránky závisí na typu proudu publikací:

- Dělící se na svazky – Je vypsán přehled všech svazků s možností jejich výběru. Příklad takového přehledu můžeme vidět na obrázku 14. Za přehledem následuje sekce se statistikami publikací vybraných svazků a sekce s grafem spoluautorství a tabulkou všech autorů vybraných svazků.
- Obsahující publikace představující svazky – Je možné si vybrat mezi zobrazením metadat publikací představujících svazky a zobrazením metadat publikací vybraných svazků. V obou případech je zobrazena sekce se statistikami publikací a sekce s grafem spoluautorství a tabulkou všech autorů.

- Nedělící se na svazky – Je zobrazena sekce se statistikami publikací a sekce s grafem spoluautorství a tabulkou všech autorů.

Graf spoluautorství je popsán v kapitole 8.7. V tabulkách autorů je u každého autora uveden celkový počet jeho publikací z daného proudu publikací nebo svazku a počet společných spoluautorů těchto publikací. Metadata každého svazku si můžeme zobrazit odděleně na samostatné stránce. Taková stránka opět obsahuje sekci se statistikami publikací daného svazku a sekci s grafem spoluautorství (viz kapitola 8.7) a tabulkou všech autorů daného svazku.



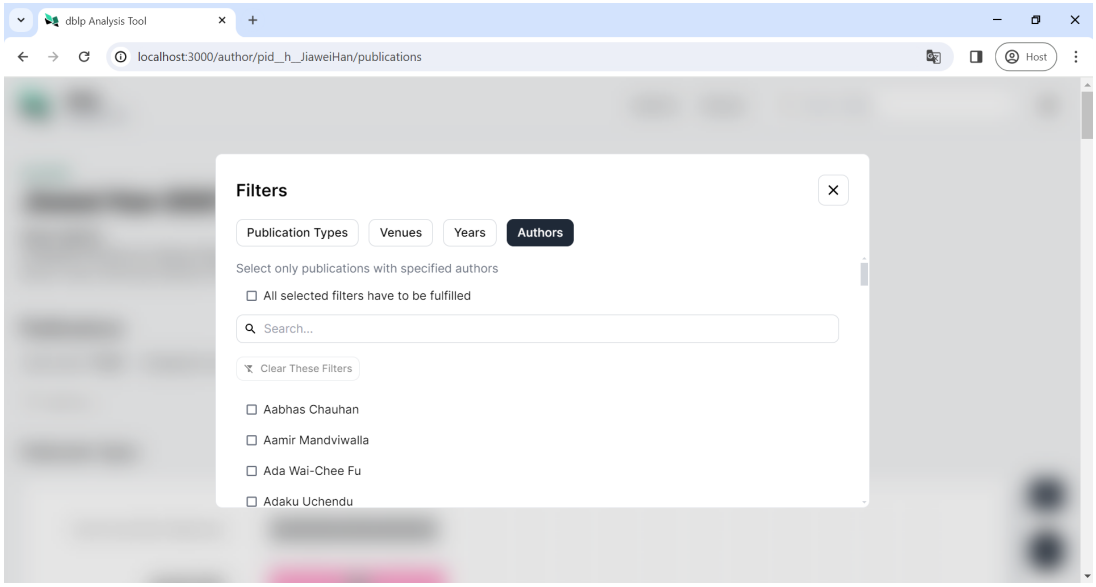
Obrázek 14: Přehled všech svazků proudu publikací s možností jejich výběru

8.6 Stránky publikací a filtrování

Pro přehled všech publikací autora, skupiny autorů nebo proudu publikací nabízí aplikace samostatné stránky. Na těchto stránkách najdeme kromě výpisu všech publikací i stejné statistiky publikací jako na odpovídajících stránkách autora, skupiny autorů nebo proudu publikací. Zobrazené publikace můžeme v aplikaci filtrovat podle čtyř různých typů filtrů:

- typ publikace,
- rok vydání publikace,
- proud publikací, do kterého je publikace zařazena,
- autor publikace.

Pokud vybereme více filtrů stejného typu, pak musí vybraná publikace splňovat alespoň jeden z těchto filtrů. U autorů si však můžeme vynutit, že má publikace splňovat všechny filtry (tedy obsahovat všechny zvolené autory). Pokud vybereme dva filtry různých typů, pak musí vybraná publikace splňovat oba filtry zároveň. Jestliže tedy vybereme například filtry pro rok „2000“ a „2001“ a pro typ publikace „Journal Articles“, jsou zobrazeny pouze články vydané v roce 2000 nebo 2001. Dialog pro výběr filtrů můžeme vidět na obrázku 15.



Obrázek 15: Dialog pro výběr filtrů publikací

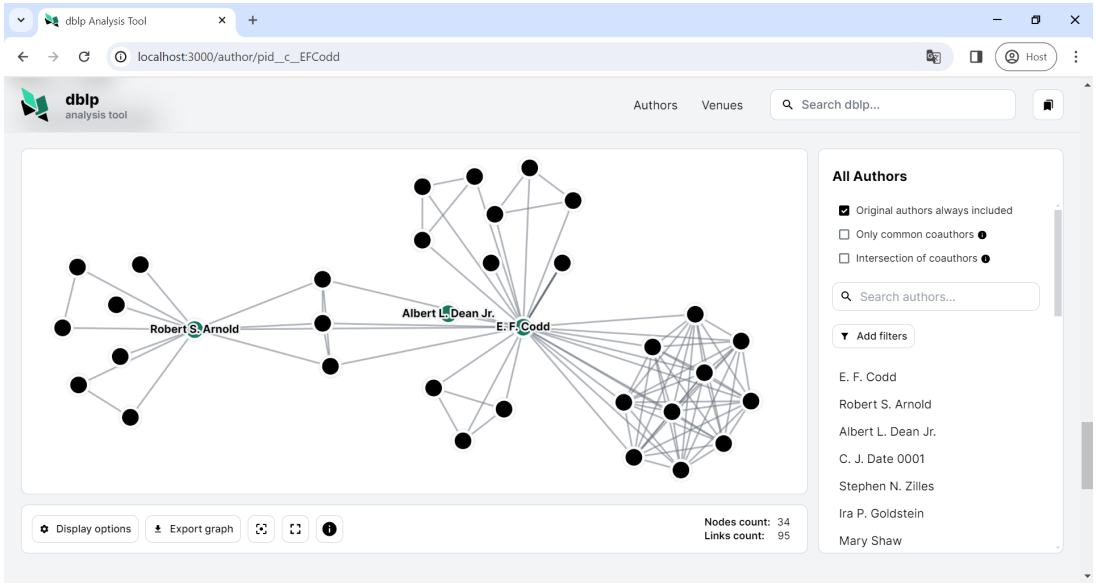
8.7 Graf spoluautorství

Graf spoluautorství je vždy tvořen všemi autory publikací z nějaké *množiny publikací*. V případě stránky autora nebo skupiny autorů se jedná o množinu publikací všech originálních autorů. *Originální autor* je autor, jehož všechny publikace jsou zahrnuty do množiny publikací. Aktuálně zobrazený autor na stránce autora a všichni vybraní členové skupiny autorů na stránce skupiny autorů jsou vždy považováni za originální autory. Další originální autory lze zařadit manuálně. V případě stránky proudu publikací je množina tvořena buď publikacemi daného proudu, nebo publikacemi vybraných svazků.

Uzly grafu představují autory publikací z dané množiny publikací. Neorientované hrany grafu představují relaci spoluautorství. Mezi dvěma autory existuje hrana pouze v případě, že jsou spoluautory stejné publikace z dané množiny publikací. Může tak nastat situace, kdy jsou dva autoři spoluautory stejné publikace, ale v grafu mezi nimi není hrana, protože tato publikace není zahrnuta v dané množině publikací. Ohodnocení hran představuje počet společných publikací dvou autorů z dané množiny publikací.

Ukázku komponenty grafu můžeme vidět na obrázku 16. Jestliže graf obsahuje dva a více originálních autorů, můžeme si nechat zobrazit pouze jejich společné spoluautory. Autory lze v grafu taktéž vyhledávat a filtrovat. Filtrování funguje obdobně jako v případě publikací. Můžeme si díky němu zobrazit pouze:

- vybrané autory,
- autory určitého typu publikace,
- autory publikace vydané v určitém roce,
- autory publikace zařazené do určitého proudu publikací.



Obrázek 16: Komponenta grafu spoluautorství

Komponentě grafu můžeme nastavit různé možnosti zobrazení, jako například možnosti zobrazení odfiltrovaných autorů, popisků uzlů nebo ohodnocení hran. Pro pohodlnější práci s grafem si navíc můžeme jeho komponentu nechat roztáhnout na celou obrazovku. Graf je možné exportovat do řady různých formátů:

- JSON,
- CSV – buď jako seznam hran, nebo jako matici (u grafů s nejvýše 4000 uzly),
- GraphViz DOT,
- GDF,

- GML,
- GraphML,
- GEXF.

Na každého autora v grafu je možné kliknout a nechat si tak zobrazit přehled všech jeho spoluautorů. Tito spoluautoři jsou rozdělení do dvou skupin – spoluautoři, kteří jsou společní s nějakým originálním autorem, a spoluautoři, kteří nejsou. U spoluautorů jsou zobrazena ohodnocení případných hran, tedy počty společných publikací z dané množiny publikací. Ohodnocení hrany se zobrazí, i pokud na ni v grafu najedeme myší.

Závěr

Výsledkem této bakalářské práce je webová aplikace, která umožňuje provádět analýzu nad daty z DBLP databáze. Aplikace plní svůj účel. Nabízí řadu vizualizací různých statistik a vztahů mezi daty. Zvlášť hodnotná může být pro uživatele komponenta grafu spoluautorství umožňující snazší pochopení vztahů mezi jednotlivými autory. Uživatelské rozhraní aplikace je přehledné a přívětivé, čímž taktéž napomáhá k snadnějšímu pochopení analyzovaných dat. Všechna analyzovaná data má uživatel možnost si stáhnout. Nejen architekturu aplikace, ale i její uživatelské rozhraní jsem navrhl tak, aby případné rozšíření aplikace o nové funkcionality bylo co nejjednodušší.

Conclusions

The result of this bachelor's thesis is a web application that allows for analysis of data from the DBLP database. The application fulfills its purpose. It offers a range of visualizations of various statistics and relationships between data. Particularly valuable for users may be the co-authorship graph component that enables easier understanding of the relationships between individual authors. The user interface of the application is clear and user-friendly, which also helps to more easily understand the analyzed data. Users have the option to download the analyzed data. I designed not only the application's architecture but also its user interface to make any potential expansion of the application with new functionalities as simple as possible.

A Zprovozněná aplikace

Pro účely testování je vytvořená aplikace v době odevzdání této práce zprovozněna na adrese http://158.194.92.113/, případně je možné ji zprovoznit z přiložených zdrojových kódů.

B Obsah elektronických dat

`text/`
: Adresář obsahující text bakalářské práce ve formátu PDF a zdrojové kódy tohoto textu.

`src/`
: Adresář obsahující zdrojové kódy aplikace.

`README.txt`
: Textový soubor obsahující postup zprovoznění aplikace.

Literatura

[1] Ackermann, Marcel R. *7 million publications* [online]. 2024 [cit. 2024-3-19]. Dostupný z: ⟨https://blog.dblp.org/2024/01/01/7-million-publications/⟩.

[2] *Under what license is the data from dblp released.* [online]. 2024 [cit. 2024-3-19]. Dostupný z: ⟨https://dblp.org/faq/1474677.html⟩.

[3] *OpenAlex.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://openalex.org/⟩.

[4] *Crossref.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.crossref.org/⟩.

[5] *OpenCitations.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://opencitations.net/⟩.

[6] *Semantic Scholar.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.semanticscholar.org/⟩.

[7] *Michael Ley.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://dblp.org/pid/00/1.html⟩.

[8] *Which technology does dblp use for searching the website?* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://dblp.org/faq/11960371.html⟩.

[9] *How to use the dblp search?* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://dblp.org/faq/1474589.html⟩.

[10] *What do I find in dblp.xml?* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://dblp.org/faq/16154937.html⟩.

[11] Beckermann, Benedikt Maria. *Dataset publications in dblp* [online]. 2023 [cit. 2024-3-19]. Dostupný z: ⟨https://www.dagstuhl.de/en/dblp/news/2023/dataset-publications-in-dblp⟩.

[12] *Am I allowed to crawl the dblp website?* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://dblp.org/faq/1474706.html⟩.

[13] *Next.js dokumentace.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://nextjs.org/docs⟩.

[14] *React.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://react.dev/⟩.

[15] *TypeScript.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.typescriptlang.org/⟩.

[16] *usehooks-ts.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://usehooks-ts.com/⟩.

[17] *server-only.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.npmjs.com/package/server-only⟩.

[18] *MongoDB.* [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.mongodb.com/⟩.

[19] *Mongoose*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://mongoosejs.com/⟩.

[20] *SWR*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://swr.vercel.app/⟩.

[21] *Cheerio*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://cheerio.js.org/⟩.

[22] *he*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.npmjs.com/package/he⟩.

[23] *uuid*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.npmjs.com/package/uuid⟩.

[24] *D3.js*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://d3js.org/⟩.

[25] *Tailwind CSS*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://tailwindcss.com/⟩.

[26] *PostCSS*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://postcss.org/⟩.

[27] *Autoprefixer*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.npmjs.com/package/autoprefixer⟩.

[28] *tailwind-merge*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.npmjs.com/package/tailwind-merge⟩.

[29] *clsx*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://www.npmjs.com/package/clsx⟩.

[30] *Class Variance Authority*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://cva.style/docs⟩.

[31] *React Icons*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://react-icons.github.io/react-icons/⟩.

[32] *Jest*. [online]. [cit. 2024-3-19]. Dostupný z: ⟨https://jestjs.io/⟩.

[33] *d3-force*. [online]. [cit. 2024-3-20]. Dostupný z: ⟨https://d3js.org/d3-force⟩.

[34] *Web Workers API*. [online]. [cit. 2024-3-20]. Dostupný z: ⟨https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API⟩.

[35] *Install MongoDB Community Edition*. [online]. [cit. 2024-3-21]. Dostupný z: ⟨https://www.mongodb.com/docs/manual/administration/install-community/⟩.

[36] *Node.js*. [online]. [cit. 2024-4-5]. Dostupný z: ⟨https://nodejs.org/⟩.