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

Bakalářská práce

Automatizované generování seznamu publikací z elektronických zdrojů

Plzeň 2022 Petr Urban

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Petr URBAN**
Osobní číslo:	**A19B0218P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Automatizované generování seznamu publikací z elektronických zdrojů**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s principy a technologiemi webových služeb, datovými strukturami a nástroji pro správu bibliografických záznamů či seznamů publikací.
2. Analyzujte možnosti automatizovaného získávání a aktualizace bibliografických dat z vybraných systémů.
3. Navrhněte webový nástroj pro sběr, správu a prezentaci publikačních záznamů osob a skupin s (polo)automatickým importem dat z externích zdrojů, zvažte přitom možnosti jeho integrace do stávajících webových stránek.
4. Nástroj implementujte s využitím vhodných technologií, ověřte jeho správnou funkčnost a uživatelskou použitelnost. Realizaci proveďte s ohledem na další vývoj a dostatečně dokumentujte.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Přemysl Brada, MSc., Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

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

V Plzni dne 4. května 2022

Petr Urban

Abstract

The goal of this bachelor thesis was to get acquainted with the principles and technologies of web services, and with the data structures and applications for manipulating bibliographic records or lists of publications. Consequently, based on this knowledge, a web application was created which communicates with a selected bibliographic database (DBLP). The application is able to represent obtained records and is integrated into the current system of the Department of Computer Science and Engineering. The implementation is designed to be extensible as more work on this topic could be exptected in the future.

Keywords: lists of publications, DBLP, web services, bibliography, bibliographic records, web application, PHP, MySQL database

Abstrakt

Cílem této práce bylo seznámení se s principy a technologiemi webových služeb, datovými strukturami a nástroji pro správu bibliografických záznamů či seznamů publikací. Na základě těchto znalostí byla vytvořena webová aplikace, která komunikuje s vhodnou bibliografickou databází (DBLP). Aplikace je schopna získané záznamy vhodně zobrazit a je integrovaná do stávajícího systému katedry KIV. Při vývoji byl brán ohled na budoucí rozšiřitelnost.

Klíčová slova: seznamy publikací, DBLP, webové služby, bibliografie, bibliografické záznamy, webová aplikace, PHP, MySQL databáze

Poděkování

Rád bych tímto poděkoval panu Doc. Ing. Přemyslu Bradovi MSc., Ph.D. za velice užitečné rady, vstřícný přístup a vedení mé bakalářské práce.

Obsah

1 Úvod

V dnešní digitální době je cíleno na ukládání veškerých dat do elektronické podoby. Těmito daty jsou myšleny mimo jiné také vědecké publikace osob pracujících například v akademických sférách.

Je velice důležité tyto publikace uchovávat na konkrétních místech, odkud budou přístupné i širokému okolí a mohou být využity v jiných pracích nebo mohou zkrátka sloužit jako inspirace pro jiné. Z tohoto důvodu vznikly bibliografické databáze uchovávající tyto informace, které je poté možné vyhledávat pomocí konkrétních klíčových slov. Takovou databází je například aplikace **DBLP** nebo **Google Scholar**. Pro jednoduchou manipulaci s těmito aplikacemi je zřízeno uživatelské rozhraní komunikující využitím protokolu HTTP(S). Uživatelům jsou poté data zobrazena v čitelné podobě, například v podobě tabulky.

Ve většině případů ale není žádoucí využívat pouze tyto aplikace samotné kvůli omezené funkčnosti. Z tohoto důvodu valná většina těchto bibliografických databází také mimo jiné nabízí webové služby, ke kterým je zřízen přístup skrze implementované endpointy na konkrétních URI. Po vytvoření požadavku na danou webovou službu je poté přenesena množina požadovaných dat na aplikaci, ze které byl požadavek vytvořen. Tyto data poté mohou být libovolně zpracována a může s nimi být libovolně manipulováno dle konkrétních požadavků.

Cílem této práce je vytvoření webové aplikace, která je schopna komunikovat s vhodnou bibliografickou databází pro získávání citačních záznamů publikací vědeckých pracovníků, a to zejména pracovníků katedry KIV. Následně musí být schopna tyto citační záznamy publikací zobrazit do vhodné podoby, například tabulky, a bude umožňovat jednotlivým uživatelům manipulaci s nimi. Manipulací je myšleno ukládání získaných publikací do své sbírky a následně i jejich mazání. Nejdůležitější funkcionalitou aplikace je integrace do stávajícího systému KIV. Ta je řešena prostřednictvím implementované webové služby, která vrací seznamy konkrétních publikací.

V první části textu jsou představeny webové služby a jejich využití. Následně jsou představeny i bibliografické databáze a nástroje pro samotnou správu těchto záznamů. Ve druhé části práce je podrobně rozepsán návrh a řešení webové aplikace. Závěr se zabývá ověřením správnosti funkcionality aplikace a její integrací do stávajícího systému katedry KIV.

2 Webové služby

2.1 Definování webové služby

Webová služba [18] je softwarový systém podporující interakci dvou strojů, resp. interakci mezi klientem a serverem, a to skrze síťové připojení. Je popsána ve strojově zpracovatelném formátu WSDL, což je protokol pro popis funkcí nabízených konkrétní webovou službou a zároveň i výstupů těchto funkcí. Webová služba může komunikovat například starším protokolem SOAP, jenž je také popsán ve formátu WSDL. Tento popis je obecně ve formátu XML.

U SOAP protokolu je v definici užito minulého času, jelikož už neplatí, že by byl SOAP standardem ve webových službách. V dnešní době existuje spousta dalších a rozšířenějších protokolů, jenž jsou užity ke komunikaci s webovou službou jako takovou. To lze například vypozorovat od WSDL verze 2.0, kdy byl umožněn popis **REST**[1] služeb.

Stejně jako ve své diplomové práci zmínil kolega Zdeněk Valeš [17], výše zmíněná definice je velice svazující i na užití pouze konkrétního formátu XML pro přenos dat mezi dvěma stroji při vzrostlé popularitě jiných jazyků jako je například **JSON**[2] nebo **YAML**[3]. Není výjimkou, když nějaké webové služby využívají těchto formátů pro přenos více dat najednou. Díky tomu je zajištěna větší kompatibilita s více aplikacemi najednou, kdy je možné si vybrat vhodný formát, který by z dané webové služby přijaly a zpracovaly dle svého uvážení, viz webové služby poskytované z **DBLP**[4]. Tato databáze je popsána v následující kapitole a zároveň je užita i jako hlavní zdroj pro získávání dat z přístupového bodu poskytovaného danou webovou službou. Tato data jsou zpracována v implementované aplikaci, která je důkladně popsána v následující kapitole **5**.

Velikou nevýhodou u webových služeb pracujících pouze s jedním konkrétním protokolem pro přenos dat spočívá zejména v nekompatibilitě v případě změny WSDL, který popisuje, jak vypadá formát přenosu dat. Na tento fakt reagují nové protokoly, které jsou vytvořeny tak, aby v případě změny v implementaci neovlivňovaly ostatní aplikace, které danou službu mohou využívat.

[1] Representational State Transfer
[2] JavaScript Object Notation
[3] YAML Ain't Markup Language
[4] Database systems and Logic Programming

2.2 Webové služby - základní pojmy

Pod pojmem `webová služba` [19] by si měl každý vývojář, jenž chce nějakou webovou službu využívat například ve své aplikaci, představit několik základních pojmů:

- **producent služby**: Softwarová nebo hardwarová část poskytující specifické služby (v dnešní době nejčastěji **server**), kde specifickými službami si lze představit nejčastěji sadu dat předávaných skrze HTTP protokol v konkrétním formátu (YAML, XML, JSON, ...),
- **konzument služby**: klient (v dnešní době nejčastěji **aplikace**), využívající služby producenta,
- **vstupní bod aplikace**: Také jinak řečeno `endpoint`, skrze který je možné získat data v konkrétním formátu, který poskytuje daná webová služba. Tento vstupní bod aplikace je přístupný například pomocí URI a HTTP protokolu,
- **API**: Již konkrétní rozhraní webové služby poskytované externím aplikacím. Toto rozhraní definuje, jak konzument může přistupovat k danému producentovi.

Hlavním přínosem webových služeb je přijetí jejich protokolů jako standardů mezisystémové komunikace, které tak vytvářejí vhodný nástroj pro komunikaci mezi firemními systémy a tím i zefektivnění Business-to-Business operací.

RPC

Remote Procedure Call [15] se silně váže s webovou službou jako takovou. RPC obecně představuje způsob meziprocesové komunikace, kdy jeden program může volat vzdálené metody jiného programu, který nemusí být vůbec součástí daného stroje, ze kterého volání probíhá. Například klient vyvolá požadavek v aplikaci, kterou aktuálně používá, a očekává nějaký výsledek. Tento program pro úspěšné získání výsledku ale využije RPC - zavolá vzdáleně metodu jiného programu skrze určitý protokol, například HTTP. To je znakově orientovaný protokol a umožňuje tak přenos dat interpretovaných například v XML, JSON nebo plain text formátu. Data se takto přenesou na konkrétní cílové zařízení, které daný protokol zpracuje. Poté na základě obsahu dat předaných skrze protokol vyvolá požadovanou metodu včetně předaných parametrů a navrátí zpět tím samým způsobem výsledek klientovi.

Endpoint

Některé aplikace, jenž mají vlastní UI a poskytují určité služby (například aplikace zobrazující aktuální počasí v grafickém režimu) mohou taktéž nabízet i vlastní webové služby, které jsou přístupné z tzv. endpointů pro externí aplikace. Rozdíl mezi normální aplikací a endpointem u webové služby je ten, že z endpointu jsou očekávaná surová data např. v právě jednom z výše zmíněných transportních formátů dat (XML, JSON, ...), jenž jsou snadno zpracovatelné v jiných strojích, které tyto data získají. Díky tomuto přímému přístupu skrze endpoint můžeme zajistit distribuci konkrétních dat do různých zařízení, kde můžou být data zpracována vlastními způsoby, například widgety v mobilních zařízení (počasí - data jsou získána z externího zdroje a zde pouze zobrazena), nebo DBLP bibliografická databáze vědeckých článků, která poskytuje hned tři endpointy pro tři různé druhy dat, které budou navrácena v konkrétním formátu (XML, JSON, BibTex, ...), jako jsou například jména jednotlivých autorů, jejich publikace atp.

2.3 Technologie webových služeb

Distribuované [19] systémy se skládají z rozmanitých a diskrétních softwarových nebo hardwarových komponent, jenž musejí být schopny spolupracovat mezi sebou pro vykonání určitého úkonu. Tyto komponenty jsou propojeny sítí a komunikace mezi nimi probíhá pomocí zasílání zpráv. Ty jsou zasílány nejčastěji použitím protokolu HTTP(S). Komponenty nemusí komunikovat mezi sebou pouze pro vykonání konkrétního úkonu, ale může díky nim být zřízena koordinace používání sdílených zdrojů mezi jednotlivými uživateli a poskytovat jim podporu v podobě komunikačního nástroje.

Hlavní charakteristikou distribuovaných systémů je absence sdílené paměti, což vede k faktu, že každá komponenta používá svou vlastní paměť a události v systému nastávají asynchronně. Toto chování může mít za negativní důsledek například zvýšenou latenci při zasílání dat po síti. Na druhou stranu to má také veliké výhody, a to zejména při výpadku jedné nebo více komponent, kdy tak nemusí dojít k selhání celého systému, protože se zachová jeho funkčnost a jednotlivé úkoly se přerozdělí mezi ostatní komponenty.

2.4 Webové služby a architektonické styly

2.4.1 SOAP (Simple Object Access Protocol)

U webových aplikací je velice důležitá komunikace skrze Internet. [12] Nejlepším způsobem jak tomu docílit je HTTP(S) protokol, jenž je podporován všemi internetovými prohlížeči a servery. Proto byl vyvinut SOAP protokol, který je popisován formátem XML, je platformně nezávislý a popisuje samotný způsob komunikace mezi jednotlivými aplikacemi.

2.4.2 SOA (Service-Oriented Architecture)

Softwarová architektura [10], která vznikla zejména pro obchodní potřeby organizací. Díky ní lze vytvořit například systém, který může být složen z několika "`microservices`", nebo jinak řečeno služeb, kooperujících mezi sebou. Všechny tyto služby dohromady tvoří jeden celý funkční systém, přičemž každá služba je oddělena od samotného jádra aplikace a může tak být snadno nahrazena jinou službou nebo může být snadno rozšířena aniž by bylo třeba jakéhokoli zásahu do samotného jádra systému využívajícího danou službu. Tato architektura se vyznačuje zejména těmito vlastnostmi:

- **logický pohled:** služba jako taková je abstraktním objektem, jenž je k dispozici v softwaru či hardwaru, nebo v databázích, a je definována podle funkcionality,
- **řízení přenášených dat:** Služba je formálně definována pomocí zpráv, které proudí mezi agenty a která tak mezi nimi může zajišťovat řízení bez ohledu na to, jak je která strana implementována, na jaké platformě a v jakém programovacím jazyce. Zároveň tato služba nepotřebuje a ani by neměla vyžadovat znalosti ohledně implementace třetích stran,
- **popisná orientace:** Služba jako taková je popsána metadaty, která jsou strojově čitelná. Tento popis plně podporuje architekturu SOAP, která říká, že by v metadatech měla být pouze data nezbytná pro správnou komunikaci mezi agenty, zbytek by neměl být viditelný,
- **granularita:** služby mají sklon k používání malého počtu operací s relativně velikými a složitými zprávami,
- **síťový přístup:** služby jsou připravené zejména k práci po síti, ačkoli to není nutný požadavek.

V této definici byl zmíněn pojem “microservices“, a to z důvodu velikého použití v dnešní době. Veliké množství rozsáhlých aplikací je rozdělených právě do těchto microservices. Aplikace poté nejsou jedním velikým monolitem, který se hůře rozšiřuje nebo opravuje.

2.4.3 COM

Dle definice Microsoftu [1] je `Component Object Model` objektově orientovaná softwarová architektura, která umožňuje sestavení aplikací z binárních softwarových komponent. Jednotlivé komponenty dokáží komunikovat mezi sebou. S tímto nápadem přišla sama společnost Microsoft v roce 1993.

2.4.4 CORBA

Základní funkcí architektury CORBA [1] je právě podpora jazykově neutrálního transparentního použití distribuovaných objektů. To znamená, že klient může používat dostupné objekty bez ohledu na programovací jazyk, ve kterém jsou implementovány. Zároveň s tím není brán ohled na počítač, na kterém jsou dané objekty spuštěny a skrze jaký komunikační protokol jsou dostupné. Objektem se rozumí jednoznačná, zapouzdřená entita poskytující nějaké služby. Klient k těmto službám přistupuje zasíláním požadavků. Forma generování požadavků závisí na jazyce, ve kterém je klient napsán. V objektově orientovaném jazyce to může být například volání metod zástupného objektu.

Distribuované objektové systémy se musí vypořádávat s několika problémy, které mohou vznikat v průběhu přenášení jednotlivých dat. Patří mezi ně například:

- problémy způsobené prodlevou či nespolehlivostí přenosu,
- sdílená paměť mezi agentem a klientem (volající službou),
- scénáře představující chybnou manipulaci mezi objektem a klientem,
- souběžné přistupování ke vzdáleným agentům,
- nekompatibilita mezi některými systémy, zejména z důvodu špatných verzí.

Z tohoto důvodu je nutné brát v potaz, zda je konkrétní distribuovaný objektový systém implementovaný za použití *COM/CORBA* nebo technologií *webových služeb*. Ve zkratce, *SOA* a webové služby jsou jednoznačnou volbou pro aplikace, které:

- pracují na principu komunikace s využitím konkrétního protokolu využívaného pro přenos dat mezi jednotlivými stroji přes internetovou síť, kde není garantovaná rychlost ani spolehlivost přenosu,
- by byly problematické z pohledu zajištění aktuální verze pro všechny žadatele a poskytovatele v jeden okamžik,
- využívají komponenty distribuovaných systémů pracujících na různých platformách a produktech od různých výrobců,
- nebo již existující aplikace, která potřebuje být strukturovaná pro práci po síti a může být reprezentována jako webová služba.

2.4.5 REST

Tato podkapitola je věnována popisu REST, nebo také celým názvem Representational State Transfer, architektonickému stylu, který je velice oblíbený a velmi často používaný v moderních webových aplikacích, které poskytují jednoduché webové služby a zároveň nevyžadují velikou a složitou logiku při práci s jednotlivými daty.

Je zde také popsán i rozdíl mezi samotným REST a SOAP rozhraním i s možnými příklady v programovacím, resp. skriptovacím, jazyce PHP[5].

REST je architektura rozhraní, která je navržena pro distribuované prostředí. [11] Pojem REST byl představen v roce 2000 v disertační práci R. Fieldinga, spoluautora HTTP protokolu, díky čemuž je mezi samotným REST rozhraním a HTTP protokolem silná vazba.

2.4.6 REST vs SOAP

Rozhraní REST narozdíl od SOAP nevyužívá vzdáleného volání procedur, také známé jako `RPC`, ale říká přímým způsobem, jak bude s daty naloženo pomocí čtyř klíčových slov. Ta jsou definována například v HTTP protokolu využitém pro přímý přístup ke zdroji, se kterým bude navázána komunikace. Komunikace samotná poté může být stavová i bezstavová, to záleží na technologii a protokolu.

Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům, kterými mohou být například samotná data a nebo stavy aplikace, které lze také popsat konkrétními daty. Všechny tyto zdroje mají své specifické

[5]Programovací jazyk, který je využit k vývoji dynamických webových stránek, které vyžadují přímou interaktivitu mezi uživatelem a samotným serverem, na kterém daná webová aplikace běží. Například e-shopy jako takové jsou dynamické a server zajišťuje různé autorizace, autentizace, uchování informací (nákupní košík) apod.

identifikátory, jinak řečeno URI, a REST definuje čtyři základní metody pro přístup k nim. Toto je zásadní rozdíl mezi SOAP či XML-RPC protokolem, kdy komunikace není orientována datově, ale procedurálně, což znamená, že webové služby nevyužívající REST rozhraní mají přesně definovaný protokol, skrze který umožňují samotnou komunikaci s externími zdroji, a zároveň může být ke každému protokolu i procedura, která je poté zavolána a teprve ta provede požadovanou operaci.

Klíčová slova `REST` architektury

Jak již bylo zmíněno výše, `REST` implementuje 4 základní metody pro přístup a manipulaci s daty. Jednotlivé metody jsou zde v kapitole popsány.

Při každém požadavku odeslaném skrze HTTP s jakýmkoli klíčovým slovem, které je popsáno níže, server vrátí odpověď i s kódem reprezentujícím stav vykonání daného požadavku. Tyto jednotlivé kódy jsou popsány v kapitole 2.5 o HTTP(S).

`GET` (Retrieve)

Základní metoda, která se používá pro získání konkrétního zdroje. Tato metoda se používá velice často, aniž by si to běžní uživatelé uvědomili. Je to například požadavek na získání obsahu konkrétní stránky, na kterou chceme přistoupit a přečíst si její obsah - z praktického příkladu to může být například zadání požadované adresy do adresního řádku jakéhokoli webového prohlížeče, kam se zadá přesná adresa, resp. URL, a požadavek se potvrdí například stisknutím klávesy enter. V tuto chvíli se odešle na server požadavek `GET` skrze HTTP(s) protokol, který navrátí obsah celé stránky na dané adrese uživateli.

`POST` (Create)

Tato metoda slouží pro přímočaré vytvoření dat. Oproti metodě GET u metody POST není předem známý identifikátor, jelikož konkrétní zdroj zatím neexistuje a je k tomu nutné využít již předem domluveného `endpointu`, který slouží jako společný identifikátor.

Pro vytvoření dat je nutné zavolat zdroj s již konkrétním URI, a to za pomocí HTTP metody `POST`. Poté se v parametru status předává konkrétní text pro nově vytvořená data.

Toto je operace, která by z důvodu bezpečnosti dané aplikace, na kterou je tento požadavek odeslán, měla být důkladně ošetřena například ve formě

autentizace či autorizace, zda konkrétní požadavek od konkrétního uživatele či stroje může být vykonán.

`DELETE`

Jak již z názvu metody vyplývá, je tímto způsobem umožněno mazání konkrétního zdroje. Jak ale zmínil i sám autor [11], může být tato operace v praxi problematická z důvodu omezení jednotlivých HTTP či HTML nástrojů, což způsobilo nahrazení volání samotné metody `DELETE` prostou modifikaci např. metody `POST` s parametrem indikujícím odstranění daného zdroje, například:

`http://server.com/statuses/destroy/zdroj.format`.

Stejně jako u metody `POST` je i zde nutné dbát zvýšené opatrnosti a operaci neumožnit komukoli, což znamená, že by daná operace měla být ošetřena například ve formě autorizace.

`PUT` (Update)

Tato metoda by se dala popsat velice obdobně jako samotná metoda `POST` s rozdílem, kdy je tentokrát předpoklad již existujícího zdroje, tudíž zde existuje přesné URI, na kterém dojde k požadované modifikaci.

Bohužel i tato metoda je stejně jako výše zmíněný `DELETE` omezena na nástroje, které jsou využity pro konkrétní komunikaci se zdrojem a platí i zde tedy využití různých náhrad pro zavolání této metody. Zároveň i u této operace by měl být také brán zřetel na bezpečnost při manipulaci s daty.

2.4.7 Vztah mezi World Wide Web a REST architekturou

World Wide Web pracuje jako síťový informační systém, jenž lze popsat v několika krocích. Agent identifikuje objekt v systému, jinak řečeno zdroj, pomocí uniformního zdrojového identifikátoru (*URIs)*. Komunikují skrze specifické datové zdroje známé po celém světě (XML, HTML, CSS, aj.), které mezi sebou vyměňují pomocí protokolů používající URIs pro zjištění přímých nebo nepřímých adres jednotlivých agentů a zdrojů.

`REST` je architektura vhodná pro spolehlivé webové aplikace, avšak je omezenější než samotná WWW architektura. `REST` Web je podmnožinou WWW (založeno na protokolu HTTP), ve kterém agenti komunikují skrze uniformní sémantické rozhraní, resp. se skládá ze 4 výše zmíněných hlavních klíčových slov: `GET`, `POST`, `DELETE` a `PUT`. Zároveň REST architektura může

být i takzvaně bezstavová, což znamená, že konkrétní zpráva je odeslána žadateli bez ohledu na to, jak dopadly již možné odeslané předchozí zprávy či jak samotné odeslání dopadne.

Lze rozeznávat mezi dvěma hlavními třídami webových služeb:

1. REST-compliant webové služby, který má za hlavní úkol manipulaci s XML, jenž reprezentuje zdroje webové aplikace, a je zde použit přístup beze stavu,

2. webová služba, ve které může být odhaleno libovolné množství operací.

Webová služba poté používá konkrétní URIs pro identifikaci zdrojů. Zároveň tak i XML datový formát pro komunikaci.

2.5 HTTP(S)

HTTP [8] (`Hypertext Transfer Protocol`) je jeden z nejpoužívanějších internetových protokolů určených pro komunikaci s WWW servery. Slouží pro přenos hypertextových dokumentů ve formátu HTML, XML, FXML a mnoha jiných typů souborů. Pro svůj přenos používá obvykle dva rezervované porty, TCP/80, kde samotný přenos probíhá nešifrovaně, a proto se začalo používat zabezpečené `TLS` spojení nad TCP. Kombinací těchto dvou spojení vznikl protokol HTTPS, který je obvykle užíván na vyhrazeném portu TCP/443.

Spolu s dalšími aplikacemi HTTP používá jednotný lokátor prostředků, tzv. **URL**[6] [2], který specifikuje umístění aktuálně hledaného zdroje v Internetu. Nejběžnějším zdrojem je webová stránka (protokol `http/https`). Nicméně se používá ale i mnoho dalších, jako je například sdílené úložiště souborů `FTP` atp.

`https://www.`⟨`název_webu`⟩`.(cz / sk / net / com / ...)`

Protokol [5] funguje na principu dotaz - odpověď, nejčastěji například pomocí internetového prohlížeče, kdy po zadání klientské žádosti do URL odešle na server dotaz ve formě plain textu, resp. čistého textu, který obsahuje informace o souboru, který chce získat (například obsah webové stránky, který je v prohlížeči zobrazen do uživatelsky čitelné podoby) a zároveň i s tím své vlastnosti, které jako internetový prohlížeč umí. Pro ukázku, jak může vypadat požadavek ze strany klienta na server:

[6] Uniform Resource Locator

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Jako první odpovědí ze strany serveru (také v podobě čistého textu) přijde informace, zda byl požadavek úspěšně vykonán či nikoli. Tato informace je reprezentována jako číslo.

Kód akce	Popis
1XX	Informační
2XX	OK
3XX	Přesměrování
4XX	Chyba ze strany klienta
5XX	Chyba ze strany serveru

Tabulka 2.1: Stavové odpovědi ze strany serveru
Zdroj: `https://www.restapitutorial.com/httpstatuscodes.html`

Protokolu `HTTP` se říká bezestavový už jen z důvodů, že nedokáže uchovávat stav komunikace a jednotlivé dotazy tak mezi sebou nemají souvislost. V případě, kdy uživatel zadá jeden a ten samý požadavek několikrát za sebou, tento dotaz se vykoná tolikrát, kolikrát to uživatel zadá, protože to server nedokáže rozeznat.

Název operace	Popis → vše na konkrétní URI
`GET`	Získání zdroje
`PUT`	Aktualizace stávající nebo vytvoření nové entity
`DELETE`	Smazání entity (pokud existuje)
`POST`	Vložení nových dat

Tabulka 2.2: Základní metody, které jsou definované v protokolu HTTP

Jak probíhá komunikace mezi klientem a serverem skrze síť

Při komunikaci mezi dvěma vzdálenými stroji přes internet (ať už klient/server nebo klient/klient) dochází k přenosu dat skrze určité protokoly, nejběžněji HTTP(S), SMTP(S), IMAP(S) atp. Konkrétně tyto zmíněné protokoly

jsou znakově orientované a jejich provoz je směrován skrze firewallem známé, a i v základu povolené porty. Jednotlivá data jsou zabalena do konkrétního protokolu podporovaného aplikací a poté poslána po síti do cílového zařízení, kde poputují od fyzické vrsty až do aplikační, kde už jsou data zpracována (interpretována) specifickými algoritmy dané aplikace a například zobrazena uživateli do čitelné podoby.

2.6 Popis formátů dat pro přenos

YAML

`YAML` [21], resp. YAML Ain't Markup Language je formát používaný pro serializaci strukturovaných dat. Jazyk je čitelný nejen člověkem, ale i strojem. Zároveň je struktura a hierarchie dat řešena pouze pomocí předsazení. Velikou výhodou tohoto formátu je neomezená úroveň vnořování.

Řetězce mohou být uvedeny znakem dvojitých i jednoduchých uvozovek, avšak s podmínkou, že znakem, kterým daná sekvence začíná, musí také i končit.

```
string: 'Wendy''s'
unicode: "Sosa did fine.\u263A"
control: "\b1998\t1999\t2000\n"
hex esc: "\x0d\x0a is \r\n"
```

Listing 2.1: Příklad uvedení textového řetězce v `YAML` formátu

Na konkrétním příkladu je znázorněna reprezentace textového řetězce uvozeného jednoduchými uvozovkami a i zápis speciálních znaků, které nejsou součástí ASCII tabulky.

```
messages:
forFree: Nic neplatis
payPrice: Zaplatis %tolik% kc
```

Listing 2.2: Další příklad zápisu `YAML` formátu

Z tohoto příkladu lze vidět i zápis speciálních proměnných, jenž jsou poté využívány i jinými programovacími jazyky, které daný YAML formát přečtou a za definované proměnné nahradí konkrétní hodnotu.

Mimo jiné je YAML formát běžně používaný v aplikacích, které se starají o závislosti při orchestraci projektů.

JSON

`JSON` [9], resp. JavaScript Object Notation je, jak již z názvu vyplývá, Javascriptový objektový zápis. Je to způsob zápisu dat nezávislý na počítačové platformě a zároveň navrženy tak, aby mohl být využit pro přenos dat. Tato data mohou být organizovaná například v datové struktuře typu seznam nebo objekt. Vstupem může být jakákoli datová struktura (například boolean, objekt, číslo, řetězec, pole), avšak výstupem tohoto formátu je vždy řetězec, který je počítačem snadno zpracovatelný.

Tento datový formát je zásadní z pohledu webových služeb, jelikož je to struktura, kterou buďto umožňují předávat jako návratová data pro jiné externí aplikace, nebo ji samy umí zpracovávat. JSON jakožto datový formát je velice důležitý i z hlediska této práce, ve které je použit jako formát dat, které jsou navráceny z `endpointu` webové služby, kterou aplikace poskytuje.

Ačkoli by se mohlo zdát z názvu JSON, že je použitelný pouze pro jazyk JavaScript, není to zcela pravda. JSON je multiplatformní datový formát, který může sloužit pro přenos dat v libovolném programovacím nebo skriptovacím jazyce, přitom jsou data čitelná i pro člověka.

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Listing 2.3: Příklad zápisu dat v `JSON` formátu
`https://json.org/example.html`

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
```

```
13          "vOffset": 250,
14          "alignment": "center"
15      },
16      "text": {
17          "data": "Click Here",
18          "size": 36,
19          "style": "bold",
20          "name": "text1",
21          "hOffset": 250,
22          "vOffset": 100,
23          "alignment": "center",
24          "onMouseUp": "sun1.opacity = (sun1.opacity / 100) *
    90;"
25      }
26  }}
```

Listing 2.4: Příklad zápisu dat v `JSON` formátu 2
`https://json.org/example.html`

Zde jsou uvedeny dva různé druhy dat, která jsou zapsána v JSON formátu. Pro zajímavost jsou to ta samá data, ale zapsána v XML formátu viz kapitola XML níže, kde je vidět, že ačkoli XML formát se může zdát přehlednější, tak má skoro o 40 % větší režii, co se velikosti přenášených dat týče.

XML

XML [20], resp. (Extensible Markup Language) je značkovací jazyk velice podobný formátu HTML[7], ale s jedním velikým rozdílem; nejsou zde přesně daná klíčová slova jednotlivých značek, ale je zde možnost vytvoření vlastních značek dle vlastních potřeb.

Díky standardizaci formátu XML mohou být přenášena data tímto formátem na jakékoli platformě a v jakémkoli programovacím jazyce. Struktura XML dokumentu je předem dána a sestává se ze značek, proto značkovací jazyk.

Jako úplně první klíčovou informací v dokumentu je samotná deklarace XML dokumentu:

⟨`?xml version="1.0" encoding="UTF-8"?`⟩

[7]Datový formát, ve kterém je popsán obsah webových stránek. Tento obsah je následně zobrazen v internetových prohlížečích.

Význam jednotlivých klíčových slov:

- version: použítá verze XML v daném dokument,
- encoding: jaké kódování (znaková sada) je použita.

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Listing 2.5: Příklad zápisu dat v `XML` formátu `https://json.org/example.html`

Tato data jsou zobrazena výše viz **2.6**, ale ve formátu `JSON`.

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

Listing 2.6: Příklad zápisu dat v `XML` formátu `https://json.org/example.html`

Zde je znázorněn zápis stejných dat viz ukázka **Listings 2.6**, ale místo formátu JSON byl užit formát XML.

Obecně z těchto dvou zápisů lze zpozorovat fakt, že JSON formát nemá tak velikou režii na způsob reprezentace dat jako XML formát, který se

může zdát sice pro člověka čitelnější, ale vzhledem k jednotlivým zavíracím a otevíracím značkám reprezentujícím začátek a konec si nárokuje výrazně větší režii na velikost dat, která se poté přenášejí, což může mít veliký vliv na rychlost přenosu dat mezi dvěma systémy. V případě, kdy by to byla webová služba, která by poskytovala skrze XML formát veliké množství dat, mohlo by dojít z uživatelského hlediska k výraznému prodlení mezi zadáním požadavku a získáním odpovědi na něj.

3 Bibliografické databáze v počítačových vědách

3.1 Bibliografie obecně

Bibliografie [3] je systematická struktura reprezentující souhrn informací, která zobrazuje nejdůležitější informace o zdrojích, například internetového článku, knížky, citaci z časopisu a mnoho dalších. Bibliografie se váže zejména ke knihám, pokud je řeč o tištěných zdrojích.

Bibliografie jako taková může být seznam publikací na základě určitého systému (např. popisná či výčtová bibliografie).

Vznikla v 16. století, kdy již bylo velice rozšířené tištění knih a bylo třeba umět tyto knihy účinně strukturovat, aby bylo jednodušší je poté dohledat, případně jinak spravovat.

Příklad bibliografické struktury

```
@online{rest-api,
    author = {Martin Maly},
    key = {REST},
    title = {REST: architektura pro webove API},
    publisher = {zdrojak.cz},
    year = {2009},
    cited = {2022/03/06},
    note = {},
    url = {https://zdrojak.cz/clanky/rest-architektura-pro-webove-api}
}
```

Listing 3.1: Příklad zápisu metadat v bibliografickém `BibTex` formátu

Tento `BibTeX` formát představující konkrétní záznam online článku, ze kterého byla čerpána část pro tuto práci, je uložen v samostatném souboru s příponou **.bib**, ve kterém je uložena celá množina všech těchto bibliografických zápisů, a na které je poté odkazováno v textu. LaTeX díky pevně danému formátu ví, jak s jednotlivými záznamy naložit a umí z nich vytvořit citaci splňující aktuálně nastavenou normu pro daný jazyk, ve kterém je práce publikována.

Vytvořená citace z výše uvedeného BibTeX zápisu

[7] Malý, M. REST: architektura pro webové API [online]. zdrojak.cz, 2009. [cit. 2022/03/06]. Dostupné z: https://zdrojak.cz/clanky/rest-architektura-pro-webove-api.

Tato citace je automaticky generována se všemi informacemi uvedenými za jednotlivými klíčovými hodnotami. Číslo v hranatých závorkách poté představuje identifikační číslo, nebo jinak řečeno číslo v pořadí, ve kterém ho LaTeX generuje.

Jednotlivé záznamy se také liší podle typu publikace. To je důsledkem zejména metadat popisujících konkrétní publikaci. Uložená metadata knihy budou obsahovat jiné informace než metadata popisující internetový článek. Množina všech těchto záznamů se nazývá bibliografie. Typy záznamů mohou být například:

- článek,
- software,
- stať ve sborníku,
- kniha,
- kapitola v knize.

3.2 Systémy (databáze) pro správu bibliografických záznamů

Systémů pro správu jednotlivých bibliografických záznamů existuje několik, avšak všechny mají své opodstatnění. Slouží jako veliké úložiště pro různé publikace v jakékoli formě (knižní, vědecké články, aj.). Tyto zdroje[1] ve většině případů jednotlivé záznamy ukládají v bibliografické podobě z důvodu snadného zpracování při další manipulaci s těmito záznamy.

Jak již jednou bylo zmíněno v textu, bibliografická struktura je celosvětově známá a velice jednoduchá na počítačové zpracování. Z tohoto důvodu vznikly systémy (resp. nástroje), které umožňují načtení bibliografického záznamu a následně jej umí také zpracovat do čitelné podoby, např. v podobě tabulky, viz **kapitola 4**.

[1] Většinou také nazývané jako databáze. Slouží k ukládání velikého množství dat, se kterými lze později znovu manipulovat.

V případě potřeby je snadné tyto systémy dohledat na internetu např. pomocí klíčových slov "*scholarly literature websites*" s použitím vyhledávacího nástroje jako je Google.

Zde jsou uvedeny dvě množiny nástrojů pro správu bibliografických záznamů. Každá množina poskytuje stejné informace, avšak má jinou funkcionalitu:

1. nástroje pro správu bibliografických záznamů:
   - Collection of Computer Science Bibliographies,
   - DBLP,
   - Google Scholar.
2. digitální knihovny:
   - ACM Digital Library,
   - IEEE Xplore,
   - CiteeSeerX.

Digitální knihovny oproti samotným nástrojům pro správu bibliografických záznamů uchovávají celé publikace, například ve formě PDF souboru. Princip je stejný jako u běžných knihoven, ale s rozdílem reprezentování publikací, které jsou pouze v digitální podobě. Oproti tomu samotné nástroje pro správu bibliografických záznamů uchovávají pouze metadata popisující jednotlivé uložené publikace.

V této práci je dále více rozepsáno několik z těchto uvedených nástrojů (databází). Byly vybrány jen ty, které jsou relevantní vůči pracovišti KIV. Databází existuje veliké množství, ale některé neuchovávají potřebné publikace pracovníků KIV, tudíž by pro tuto práci byly nepřijatelné. Zde jsou zmíněny jen pro zajímavost.

3.2.1 DBLP

DBLP[2] je bibliografická databáze zejména v oboru počítačových věd. Dají se zde najít reference týkající se zejména vědeckých pracovníků v této oblasti.

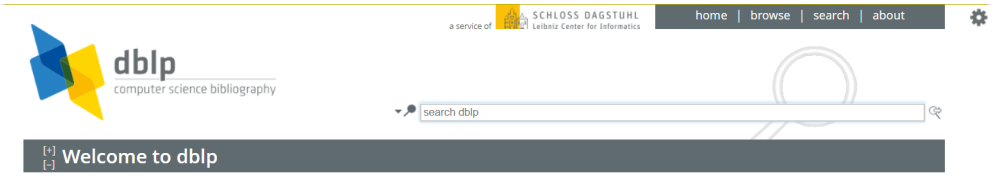
Na začátku vývoje DBLP, jenž nastal kolem roku 1993, byl tento projekt považován za malý experiment, který se díky své komfortní funkcionalitě postupem času dostal mezi nejpoužívanější a nejoblíbenější online referenční databáze na jednotlivé publikace pracovníků v oboru počítačových věd.

[2] `https://dblp.org/`

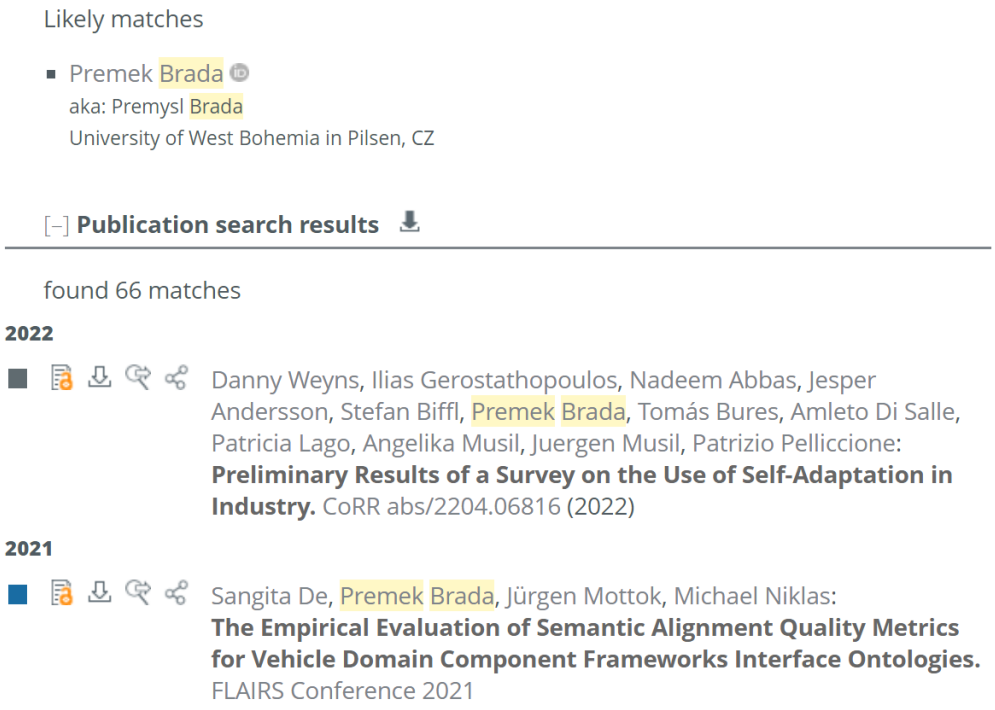
Cílem DBLP je podpora práce vědeckých pracovníků v počítačových vědách, zejména přístupem ke všem důležitým metadatům a zároveň i k odkazům na jejich originální elektronické vydání, a to zcela zdarma.

Každý autor je evidován v aplikaci unikátním identifikátorem, tzv. PID (person identifier), dle kterého je možné najít veškeré zveřejněné publikace onoho autora.

Z pohledu bibliografie je nutné podotknout nedostatek nástroje DBLP viz kapitola 3.2.1. Tím je poskytování neúplných informací (metadat) u některých záznamů, jenž jsou v DBLP databázi uloženy.



Obrázek 3.1: Základní obrazovka vyhledávacího nástroje DBLP



Obrázek 3.2: Výsledek vyhledání v DBLP

Webová služba použita v DBLP projektu

Webová služba použita v `DBLP` zahrnuje hypertextové odkazy zdrojů třetích stran. Tyto odkazy jsou použity v souladu s autorskými právy, avšak není zaručena bezchybnost informací, resp. metadat, které jsou zde uvedena, a to z několika důvodů:

- není zcela zaručena správnost informací, které byly zadány ručně uživatelem,
- uvedené odkazy třetích stran se automaticky neaktualizují, tudíž není zaručena jejich aktuálnost.

Oba dva výše zmíněné body jsou samozřejmě řešitelné. V případě neaktuálních informací lze kontaktovat `DBLP` tým a ten obnoví informace na aktuální.

Webová služba používá tři hlavní vyhledávací služby:

- `https://dblp.org/search/publ/api`
  - využíváno pro získání jednotlivých publikací
- `https://dblp.org/search/author/api`
  - využíváno pro získání všech podobných autorů, kteří se objeví ve vyhledávání
- `https://dblp.org/search/venue/api`
  - využíváno pro získání jednotlivých míst konání.

`DBLP` vyhledávací webová služba funguje na několika základních parametrech:

- `q` ⇒ reprezentující začátek databázového dotazu, který se využije pro hledání všech možných výsledků,
- `format` ⇒ jako výchozí hodnota je *XML*. Lze ho nastavit i na *JSON*. Toto je výstupní formát, ve kterém budou předány nalezená data,
- `h` ⇒ Nastavení maximálního počtu nalezených výsledků. Základně je nastaven na 30 záznamů,
- `f` ⇒ První nalezená sekvence výsledku, začínající od 0, která se bude vracet jako návratová hodnota. V kombinaci s parametrem `h` může být použit pro stránkování výsledků z vyhledávání,

- `c` ⇒ Omezení maximální hodnoty pro čas dokončení. V případě nevrácení hodnoty do určitého časového úseku bude operace pozastavena. Z důvodu šířky pásma je toto číslo v základu omezeno na 1000 záznamů.

Ukázka získávaných dat

```
<result>
  <query id="740468">Premek* Brada*</query>
  <status code="200">OK</status>
  <time unit="msecs">0.25</time>
  <completions total="1" computed="1" sent="1">
  <c sc="130" dc="65" oc="130" id="44306229">brada</c>
  </completions>
  <hits total="65" computed="65" sent="1" first="0">
     <hit score="3" id="290705">
        <info>
          <authors>
            <author pid="242/4077">Sangita De</author>
            <author pid="22/6314">Premek Brada</author>
            <author pid="16/6896">Jurgen Mottok</author>
            <author pid="242/4209">Michael Niklas</author>
          </authors>
        <title>The Empirical Evaluation of Semantic Alignment
     Quality Metrics for Vehicle Domain Component Frameworks
    Interface Ontologies.</title>
        <venue>FLAIRS Conference</venue>
        <year>2021</year>
        <type>Conference and Workshop Papers</type>
        <access>open</access>
        <key>conf/flairs/DeBMN21</key>
        <doi>10.32473/FLAIRS.V34I1.128512</doi>
        <ee>https://doi.org/10.32473/flairs.v34i1.128512</ee>
        <url>https://dblp.org/rec/conf/flairs/DeBMN21</url>
        </info>
        <url>URL#290705</url>
     </hit>
  </hits>
</result>
```

Listing 3.2: Data získána z DBLP webové služby

Tato data byla získána přes dotaz s použitím konkrétního URI:

https://dblp.org/search/publ/api?q=Premek_Brada&h=1&format=xml

Je zde složen URL dotaz, který posílá žádost na DBLP endpoint o zaslání

jednoho záznamu (pokud existuje) pro autora publikace Premek Brada a to ve formátu XML.

3.2.2 Google Scholar

Základní popis Google Scholar

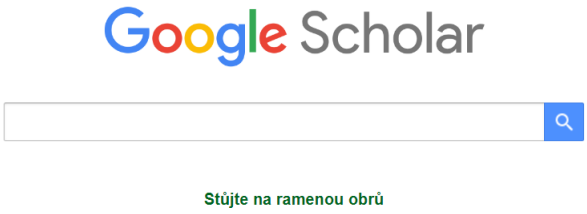
`Google Scholar`[3] je databáze uchovávající informace o veškerých publikacích, a to celosvětově, což je výrazný rozdíl oproti všem ostatním systémům. Každý uživatel registrovaný pod platformou `www.google.com` má již svůj účet i na této platformě a může jí tedy využívat.

Funkcionalita Google Scholar

Funkcionalita této platformy je vesměs velice podobná funkcionalitě vyhledávacího nástroje `www.google.com`. Uživatel pouze zadá klíč, dle kterého chce vyhledat vědecké články a stiskne příslušné tlačítko pro potvrzení požadavku.

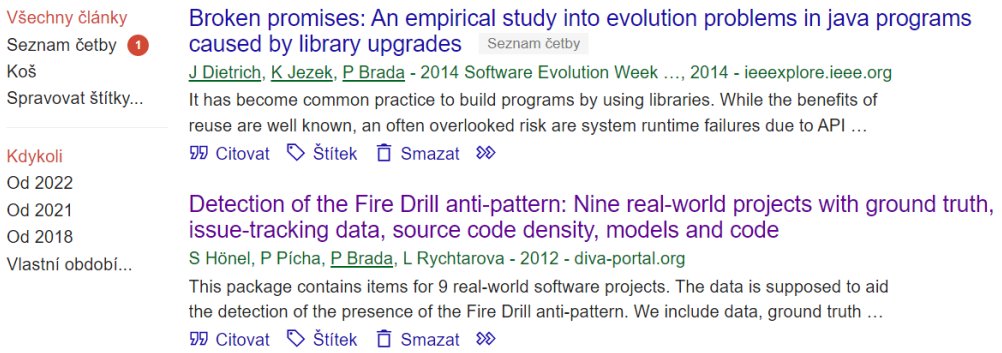
Zobrazí se veškeré výsledky, které obsahují prvky klíčového slova, dle kterého se vyhledávalo. Poté se uživatel může rozhodnout, jak tyto výsledky následně zpracovat. Je možné si je uložit do úložiště na Google Scholar platformě pomocí tlačítka `uložit`, případně je lze ocitovat stisknutím tlačítka `citovat`. Funkcionalita tlačítka `citovat` je velice abstraktní. Umožňuje ihned několik výstupních formátů, ve kterých lze tento text prezentovat na jiných uživatelem zvolených zdrojích. Těmito zdroji jsou např.: *BibTeX*, *EndNote*, a jiné.

Uložené záznamy na platformě Google Scholar lze zpětně najít na záložce *Moje knihovna*. Tyto záznamy lze mazat, upravovat, citovat je či jim přidělovat různé štítky, dle kterých je poté možné nad uloženými záznamy vytvářet filtry a nechat si tak zobrazit pouze ty záznamy, které obsahují jeden nebo více požadovaných štítků.



Obrázek 3.3: Základní obrazovka vyhledávacího nástroje Google Scholar

[3] `https://scholar.google.com/`



Obrázek 3.4: Výsledek vyhledání nástroje Google Scholar

Vyhledávací API pro Google Scholar

Společnost Google bohužel neposkytuje žádné API, které by bylo volně přístupné pro ostatní aplikace. Avšak existuje webová aplikace **SerpApi**[4], která tento problém řeší jakožto zprostředkovatel požadavků mezi platformou Google Scholar a aplikací, jenž si žádá o data. K využívání API pro získání záznamů z Google Scholaru je nutná registrace na jejich webové aplikaci a následné získání soukromého klíče, pod kterým se budou jednotlivé požadavky provádět.

Požadavky na vyhledávací API se zasílají stejně jako do aplikace DBLP, zasláním požadavku na konkrétní URL. Pro znázornění, vyhledávání může vypadat například následovně:

```
https://serpapi.com/search.json?engine=google_scholar&
q=biology&api_key=secret_api_key
```

3.2.3 ACM DL (Digital Library)

Základní popis ACM DL

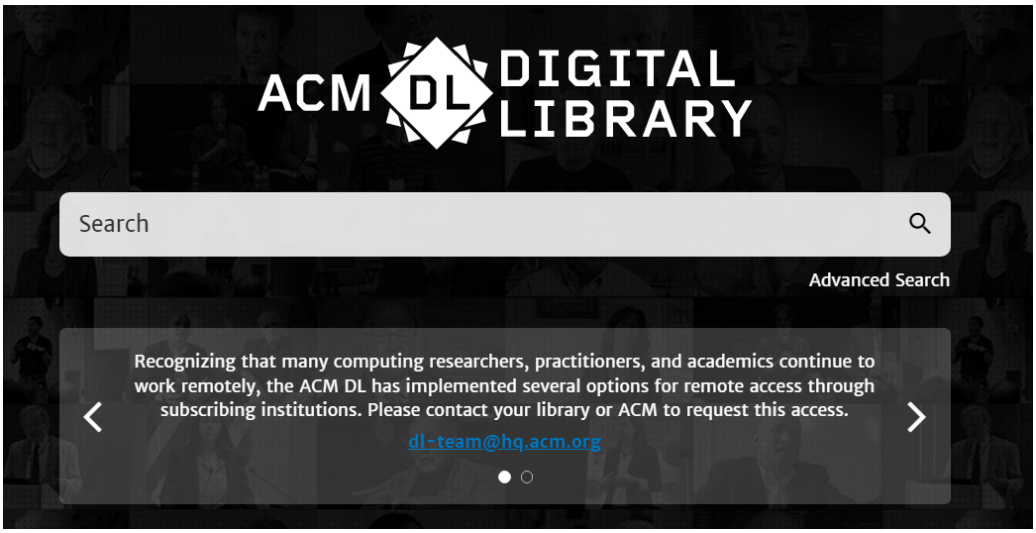
`ACM DL`[5] funguje na obdobném principu jako systém DBLP. Každý uživatel zde musí mít vlastní účet pro možnost vkládání či editaci konkrétních publikací, což oproti systému `Google Scholar` omezuje množinu uživatelů na registrované, přičemž jak bylo zmíněno výše v kapitole u `Google Scholaru` (3.2.2), tam se indexuje celý svět bez ohledu na registrované uživatele. Ve

[4] `https://serpapi.com/google-scholar-api`
[5] `https://dl.acm.org/`

chvíli, kdy je uživatel přihlášen ke svému účtu, může manipulovat s nalezenými zdroji (ukládat, mazat, upravovat, citovat aj.). Vyhledávání zdrojů je k dispozici i nepřihlášeným uživatelům a je zcela zdarma, avšak některé z funkcionalit jsou omezeny.

Funkcionalita ACM DL

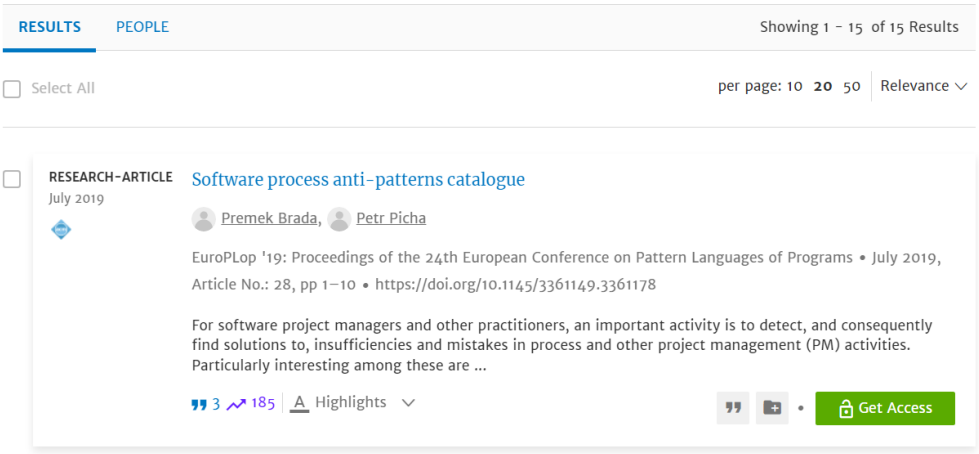
Funkcionalita tohoto nástroje je podobná nástroji `Google Scholar`, kdy uživatel zadá vyhledávaný požadavek dle určitého klíče do vyhledávací pole a stiskne příslušné tlačítko pro vyhledání. Poté bude uživateli nabídnuta množina (prázdná či neprázdná) s výsledky vyhledávání. Uživatel je může taktéž citovat na jiných zdrojích a to ihned v několika možných výstupních formátech, jako je např.: *BibTeX*, *EndNode*, aj. Avšak je také možnost celou citaci stáhnout do lokálního úložiště na počítači v onom konkrétním formátu a není nutné tento zdroj kopírovat.



Obrázek 3.5: Základní obrazovka vyhledávacího nástroje ACM DL

Vyhledávací API pro ACM DL

Z webových stránek ACM DL není patrné, že by poskytovali dokumentaci k API.



Obrázek 3.6: Výsledek vyhledávání nástroje ACM DL

3.2.4 The Collection of Computer Science Bibliographies

Základní popis CCSB

Nástroj, resp. bibliografická databáze, `CCSB`[6] se může zdát jako primitivní webová aplikace, avšak její přívětivá interaktivita s uživateli ji dělá velice užitečnou a praktickou. Jako ve výše zmíněných nástrojích se zde vyhledává pomocí klíčového slova. Je zde navíc možnost zadání dalších klíčových slov, dle kterých lze docílit přesnějšího výsledku vyhledávání.

Obrázek 3.7: Základní obrazovka systému CCSB

Jak si lze povšimnout na obrázku **3.7**, hlavní funkcionalita tohoto nástroje spočívá v zadání konkrétních klíčových slov. Výhoda zde také spočívá v jednoduchosti uživatelského rozhraní, které je velice přehledné.

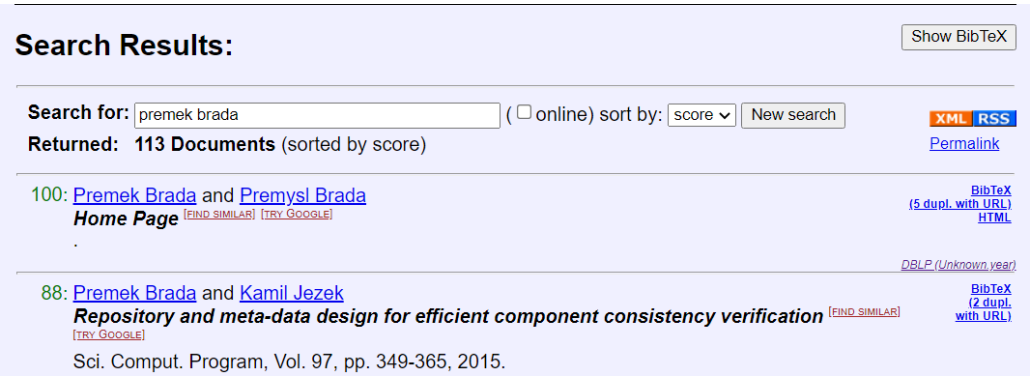
[6] `https://liinwww.ira.uka.de/bibliography/`

Základní funkcionalita CCSB

Může se zdát, že dle hypertextových odkazů na původní zdroj publikace tento nástroj úzce spolupracuje s `DBLP` nástrojem. Stejně tak má uživatel možnost vložení vlastního záznamu pomocí záložky *Add*. Výhodou je i ten fakt, že není třeba se registrovat pro manipulaci se záznamy (vložení, získání). Z pohledu bezpečnosti se jedná o riziko, že některé záznamy nemusí odpovídat skutečnosti. Při nalezení některého ze zdrojů, který se může zdát podezřelým, by na toto měl uživatel brát zřetel.

`CCSB` nástroj umožňuje také zobrazení konkrétního výsledku pomocí nástroje `Google Scholar`, na který má vygenerovaný odkaz již s uvedenými klíčovými slovy.

Veškeré nalezené zdroje lze také vložit do jiných, uživatelem zvolených nástrojů, pomocí vygenerované *BibTeX* struktury, kterou lze pouze zkopírovat a vložit.



Obrázek 3.8: Výsledky vyhledávání nástroje ccsb

Vyhledávací API pro CCSB

The Collection of Computer Science Bibliographies neposkytuje žádné API, díky kterému by bylo možné získat specifická data. Aplikace jako taková sama získává data z bibliografické databáze DBLP na základě zadaných klíčových slov, podle kterých poté hledá.

4 Nástroje pro správu bibliografických záznamů

4.1 Úvod

Nástroje pro správu bibliografických záznamů jsou důležitým prostředníkem mezi uživatelem a úložištěm, resp. databází, ze které je konkrétní bibliografie získávána. Samotná bibliografická struktura nemusí být pro všechny na první pohled zřejmá a dalo by se říci, že není uživatelsky přívětivá. Proto vznikly tyto nástroje, které díky jedinečné bibliografické syntaxi ví, jak jednotlivé záznamy uložit a jak je zpracovat do interních datových struktur, ze kterých poté tyto záznamy vykreslují do čitelné podoby, např. do tabulky.

Nástroje pro správu bibliografických záznamů nejsou určeny pouze k jednoznačnému zobrazení. Nabízí uživatelům spousty dalších možností, jak se získanými záznamy naložit. Některé dokonce kooperují s jinými nástroji, které tyto záznamy umí uložit do své databáze a uživatel je tak může mít pohodlně a kdykoli k dispozici již v čitelné podobě.

Další velikou výhodou je možnost integrace získaných záznamů z těchto nástrojů. Některé nástroje nabízejí možnost integrace do jiných zdrojů s vygenerovanou syntaxí, např. v HTML aj., které stačí pouze zkopírovat a vložit na místo určení.

4.2 Klíčová slova pro vyhledání jednotlivých nástrojů

Množina těchto vybraných nástrojů byla nalezena přes vyhledávací nástroj `www.google.com` s použitím klíčových slov: Citation plugin for academic bibliography.

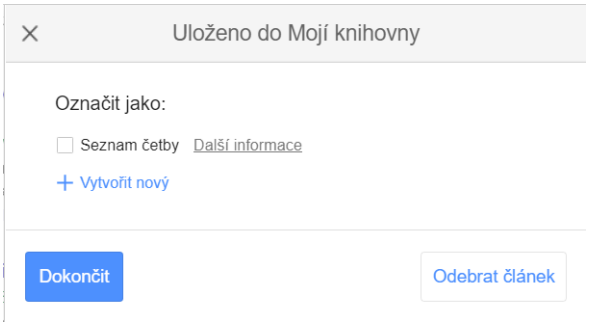
4.3 Google Scholar

`Google Scholar` je nejen samotná databáze uchovávající bibliografické reference vědeckých článků, ale také i nástroj na samotnou správu těchto záznamů.

Pomocí uživatelsky přívětivého zobrazení zpracovaných výsledků se může

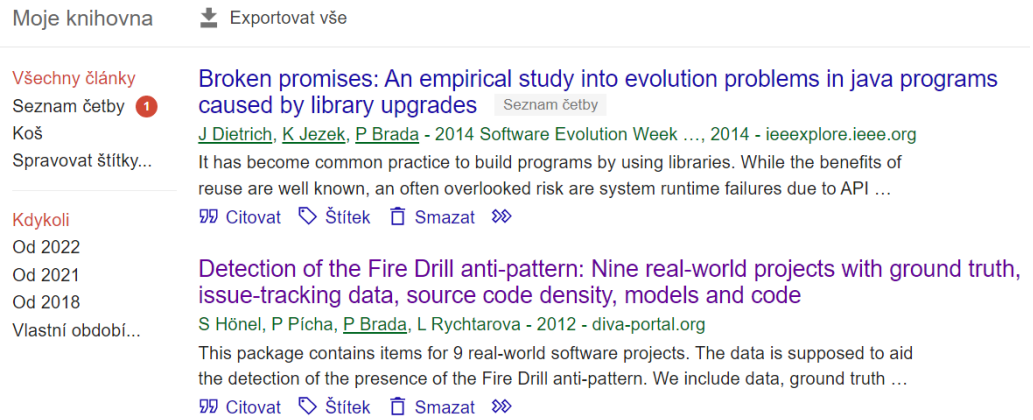
uživatel rozhodnout co dále. Může si konkrétní záznamy uložit, případně je citovat dle zvoleného formátu dále. Nástroj `Google Scholar` umožňuje vygenerování až čtyř výstupních formátů, ve kterých lze data vložit na jiné, uživatelem zvolené, zdroje.

Při zvolení možnosti *uložit* publikaci je uživateli nabídnuto z jeho veškerých kategorií, které si v průběhu času při práci s nástrojem sám vytvořil.



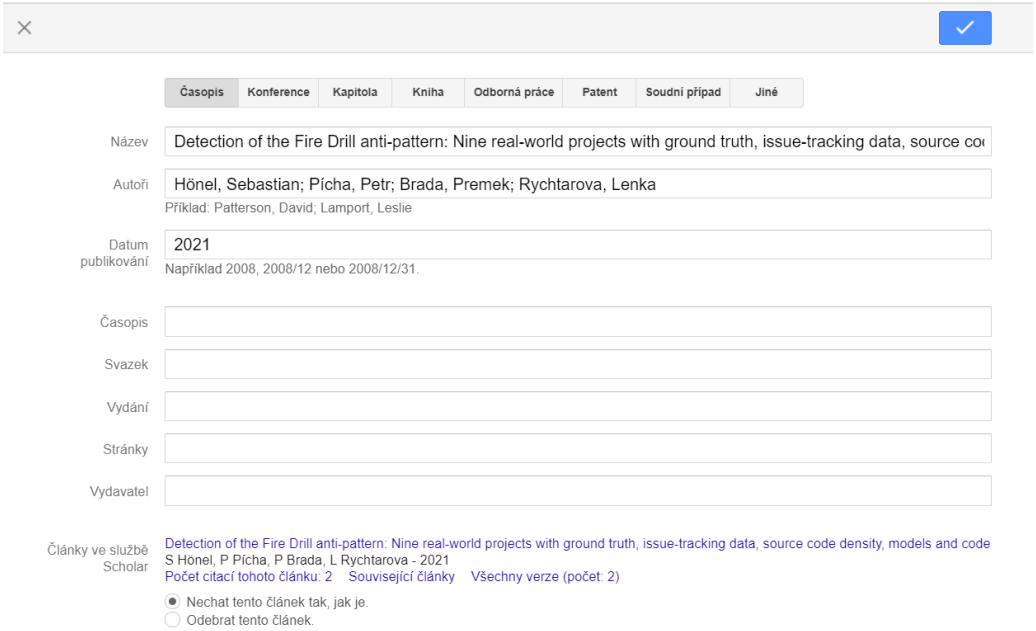
Obrázek 4.1: Uložení publikace v Google Scholar

Uživatel může zvolit i více kategorií, pod kterými bude publikace uchována. Jednotlivé kategorie si uživatel může spravovat.



Obrázek 4.2: Kategorie uložených publikací

Veškeré publikace v jednotlivých kategoriích lze také modifikovat. Avšak modifikace se projeví pouze u konkrétního uloženého záznamu, nikoli u globálního záznamu, který je nabídnut i ostatním uživatelům. Znamená to tedy, že při uložení záznamu nedochází k uložení originální publikace, ale dochází k vytvoření totožné kopie, se kterou lze libovolně manipulovat.



Obrázek 4.3: Modifikace publikace

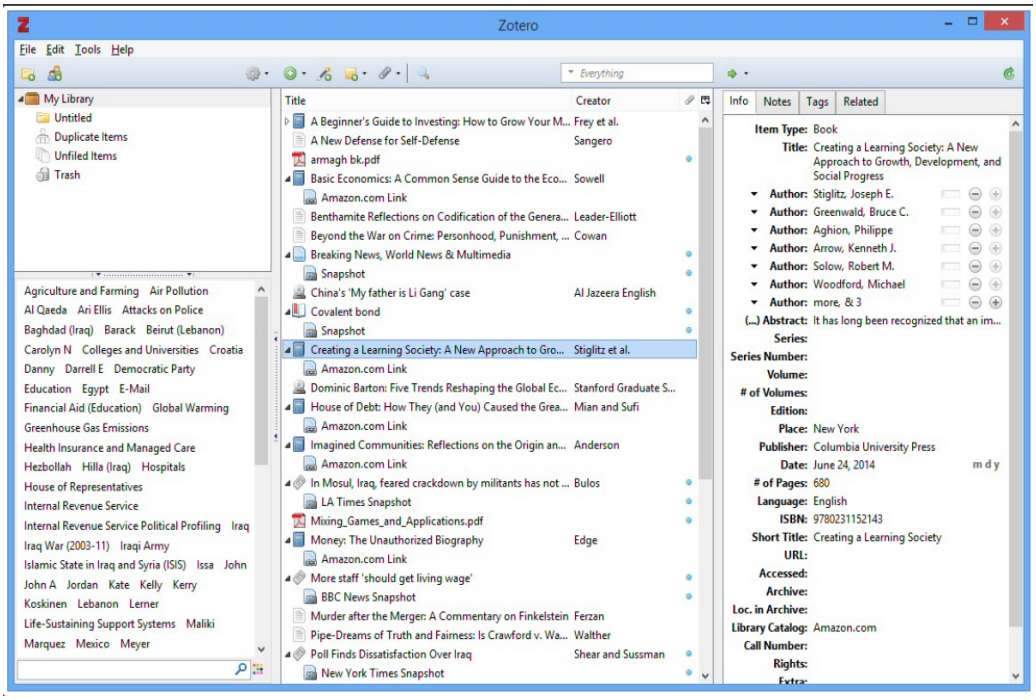
V této kapitole o nástroji `Google Scholar` byla zmíněna i možnost automatického generování citací pomocí jednoho kliknutí. Vygenerovaná citace vypadá následujícím způsobem:

Obrázek 4.4: Citace publikace

4.4 Zotero

`Zotero`[1] [6] je veřejná webová aplikace, která provozuje vlastní databázi uživatelů a jejich dat. Jednotlivá data představují zdroje ať už ve formátu bibliografickém, citačním tak i jiném. Lze toho poté využít na webových aplikacích běžících např. na platformě *WordPress*[2], kde `Zotero` funguje jako samostatný plugin, jenž se páruje pomocí veřejného klíče, který je přidělen (resp. vygenerován) na `Zotero` portále po vyžádání. Tento plugin poté generuje vlastní značky do textu, za které následně dosadí data, jenž jsou definována na `Zotero` portále pod jednotlivými klíčovými slovy.

Výhodou tohoto přístupu je automatické doplňování citací ve formátu, který si zvolil autor, jenž danou citaci poté využívá pro své účely. Jelikož se tato citace a její způsob reprezentace vyskytuje na jednom konkrétním místě, a to `Zotero` portále, lze její formát zobrazení kdykoli změnit. Tato změna se dynamicky prokáže ve všech zdrojích, kde na danou citaci pod konkrétním klíčovým slovem bylo odkazováno, což vede k veliké úspoře času.



Obrázek 4.5: Uživatelské prostředí aplikace Zotero

[1] `https://www.zotero.org/`

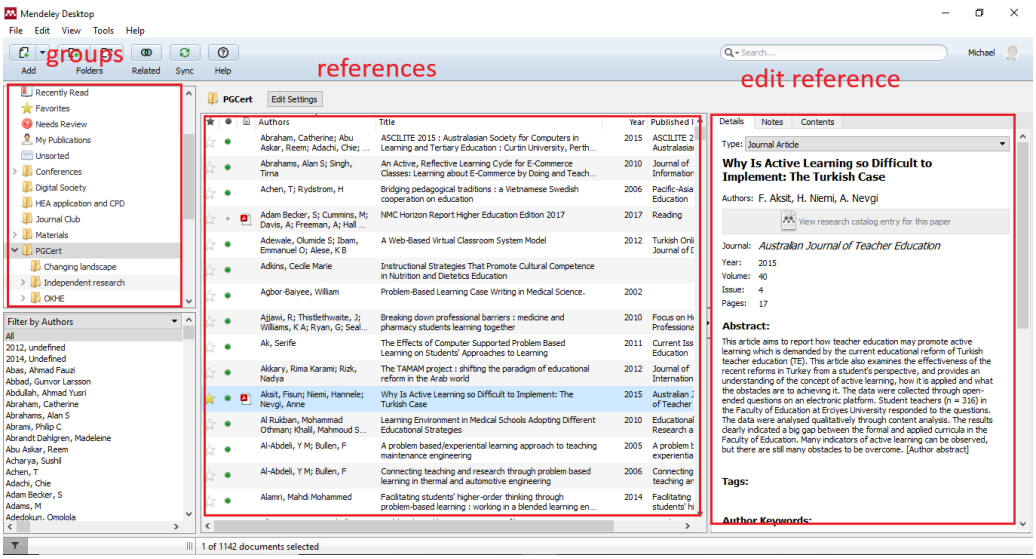
[2] Redakční systém, který umožňuje pomocí uživatelského grafického rozhraní zpracovávat a provozovat internetové stránky bez nutnosti psaní kódů

4.5 Mendeley Desktop

`Mendeley`[3] [4] je firma sídlící v Londýně, která poskytuje produkty a služby zejména pro výzkumné akademické pracovníky. Jejím neznámějším produktem je `Mendeley Dekstop`, *stand alone aplikace*[4], která umožňuje jednoduchou manipulaci s vědeckými články. To znamená jejich uložení, sdílení, upravování a tak podobně.

`Mendeley Desktop` může být propojen i s *cloudovým*[5] úložištěm, což vede k bezproblémové synchronizaci v případě změny zařízení, případně při používání jako plugin doplňku webových prohlížečů, které jsou také propojeny s online úložištěm a samozřejmě kooperuje i s počítačovou verzí.

Pro manipulaci s daty je zprostředkováno grafické uživatelské rozhraní, kde lze strukturovat jednotlivé reference do různých skupin, adresářů a tak dále. Uživatel může přidávat, organizovat, číst, zvýrazňovat a anotovat reference. Nadále mezi nimi samozřejmě může i vyhledávat a citovat je.



Obrázek 4.6: Uživatelské prostředí aplikace Mendeley Desktop
Zdroj obrázku: `https://medium.com/specialist-library-support/beginners-reference-management-with-mendeley-5b80d06a0111`

[3] `https://www.mendeley.com/guides/desktop/01-desktop-interface`
[4] Stand alone aplikace, také jinak řečeno samostatně běžící aplikace
[5] Online úložiště, do kterého se ukládají data patřící konkrétnímu uživateli; přístup k nim lze zprostředkovat z jakéhokoli zařízení pouze s použitím přístupových údajů.

4.6 Cloodo

Nástroj Cloodo[6] je v podobě pluginu do *WordPress* redakčního systému, který umožňuje vytváření seznamů publikací založených na **CSL**[7]. Umožňuje přímou integraci se sdílenými veřejnými systémy jako je *BibSonomy* nebo *PUMA*.

BibSonomy je aplikace založena na sdílení publikací. Umožňuje integraci týmově orientovaných publikací, což umožňuje správu jednotlivých referencí s použitím například filtrů podle uživatelů či výzkumných skupin.

Pro možnost použití tohoto pluginu je nutnost registrace a vytvoření si veřejného API klíče, pomocí kterého je možné propojit `Cloodo` s webovou aplikací, která tohoto pluginu využívá.

4.7 Enhanced BibliPlug

Nástroj `Enhanced BibliPlug`[8] umožňuje vytvoření repozitáře pro uschování výsledků vědeckých článků pro konkrétní akademické pracovníky ať už pro jednotlivce či celé skupiny, které na dané problematice spolupracují.

`Enhanced BibliPlug` je úzce spojen se Zotero pluginem a umožňuje tak synchronizaci s existujícím účtem na platformě Zotero, jenž se dá využít právě v tomto pluginu.

`Enhanced Bibliplug` se dá tedy shrnout jako rozšíření pro Zotero, který umožňuje až těchto sedm vlastností, jenž mohou zpříjemnit požitek uživatelů využívajících této služby:

- databázové schéma pro uchování bibliografických referencí,
- administrace pro správu jednotlivých referencí,
- *short code*[9] pro jednoduché užití referencí na základě autora, roku a typu publikace,
- možnost spojení a synchronizace se Zotero účty,

[6] `https://cloodo.com/projects/bibsonomy-puma-csl-publications-tag-cloud-widget`

[7] Jazyk reprezentující stylistiku citace (citací)

[8] `http://wordpress.alternativelist.com/plugin/enhanced-bibliplug-alternative-and-similar-wordpress-plugins-16383.html`

[9] Zkrácený kód (nebo také klíč), který se poté transparentně nahradí obsahem, jenž se ukrývá za tímto zkráceným kódem

- vlastní administrativní stránka pro zobrazení specifických dodatečných informací přihlášeného uživatele (akademický titul, vytvořené publikace aj.),
- short code pro zobrazení všech uživatelů využívajících této služby na konkrétní webové aplikaci,
- možnost seskupení referencí podle vlastních kategorií a klíčových slov.

4.8 RefWorks

Nástroj `RefWorks`[10] [7] je velice podobný nástroji `Mendeley Desktop`. Uživatelské reference jsou uložené v online databázi. `RefWorks` vznikl zejména pro pracovníky v akademické sféře a studenty a je široce podporován jinými databázovými poskytovateli bibliografických referencí, kteří umožňují přímý výstupní formát do `RefWorks` formátu. Avšak existují případy, ve kterých je nutné získanou referenci z dané databáze uložit do počítače jako prostý textový soubor a poté jej ručně importovat do `RefWorks` aplikace.

Stejně jako u `Mendeley Desktop` si uživatelé tohoto nástroje mohou spravovat jednotlivé reference pomocí velice přívětivého grafického uživatelského rozhraní, ve kterém je možné jednotlivé záznamy přidávat, prohledávat, upravovat a mazat. Jednotlivé reference lze také seskupovat do různých skupin či podskupin. Nechybí ani možnost sdílení jednotlivých referencí s vybranou množinou lidí.

`RefWorks` umožňuje také přímý import z aplikace `Mendeley Desktop` nebo jiných souborů, jenž splňují daný formát. Nechybí zde ani možnost přímého nahrání jakéhokoli dokumentu.

V roce 2005 přišli vývojáři `RefWorks` aplikace s významnou novinkou, pluginem do aplikace Microsoft Word, který umožňuje přímé nahrání `RefWorks` reference do aplikace. Ta je poté převedena do citace uvnitř textu a seznamu referencí, a to v různých variantách.

4.9 EndNote

Nástroj `EndNote`[11] je velice podobný výše zmíněnému nástroji `RefWorks`, což znamená, že jednotlivé záznamy jsou ukládány do online databáze. Stejně tak i veliké množství poskytovatelů bibliografických databází předpokládá export do výstupního formátu, na kterém si zakládá `EndNote` aplikace, což

[10] `https://www.refworks.com/refworks2/?`
[11] `https://endnote.com/`

má za důsledek ušetření uživatelského času při ukládání referencí. Stejně tak se ale vyskytují problémy u některých poskytovatelů bibliografických databázových referencí, které vyžadují export (uložení) konkrétní reference do počítače jako prostý textový soubor a následného importu do `EndNote` aplikace pro správnou funkcionalitu.

Je možné jednotlivé reference importovat (vkládat), upravovat, mazat a sdílet je s vybranými lidmi či přímo vybranou skupinou. Při vkládání reference je možné si ji strukturovat do skupin.

4.10 Srovnání RefWors, Zotero, Mendeley Desktop a EndNote nástrojů

Na webové stránce[12] se nachází tabulka se všemi čtyřmi aplikacemi, které mezi sebou porovnává. Nachází se zde nejdůležitější informace, které pomohou uživateli v rozhodnutí, který nástroj použít tak, aby sloužil nejlépe pro jeho účely.

Tabulka s porovnáním je zde alespoň částečně rozepsána a jsou zde také zmíněny nejdůležitější rozdíly mezi jednotlivými nástroji.

Zatímco nástroje `Zotero`, `Mendeley` a `EndNote` jsou zcela zdarma, tak oproti tomu je nástroj `RefWorks` nabízen zdarma pouze studentům a zaměstnancům na univerzitě Toronto (University of Toronto). Cena za tento nástroj pro širokou veřejnost zde uvedena není, ale možnost migrace do jiné instituce tu rozhodně je. Když se toto zrekapituluje, tak se dojde k závěru, že `RefWorks` je jako jediný z těchto čtyř nástrojů zdarma pouze pro studenty a zaměstnance na univerzitách, kteří mají onu licenci k dispozici od univerzity.

Tyto nástroje se naopak neliší ve způsobu, jak je lze získat. U všech je nutné vytvořit si účet na daných internetových stránkách, kde poté bude zpřístupněn odkaz ke stažení.

Všechny výše zmíněné aplikace nabízejí jednoduché ovládání skrze přívětivé uživatelské rozhraní ať už ve formě doplňku ve webovým prohlížeči, tak i jako samostatné desktopové verze[13]. Je nabídnuta i možnost přímého vkládání dat skrze určité databáze, se kterými jednotlivé nástroje umějí pracovat, což vede k velikému záporu a tím je omezená množina databází, se kterými umí spolupracovat, což není případ nástroje `Mendeley Desktop`, který pro import dat používá webový doplněk, skrze který lze ukládat jednotlivé citace publikací, které uživatel zrovna v danou chvíli chce.

[12] `https://guides.library.utoronto.ca/citationmanagement/comparison-table`

[13] Desktopová verze je v softwarové terminologii aplikace, která byla vyvinuta jako počítačová verze

4.11 Nástroje ve formě pluginů pro Google Docs

Existuje také celá řada rozšiřujících pluginů[14], jenž jsou propojeny s `Google Docs` a umožňují uživatelům komfortní práci pro rychlé spravování fererencí uvnitř textu. Tato práce se těmito doplňky hlouběji nezabývá a jsou zde uvedeny pouze pro doplnění širšího přehledu nástrojů používaných pro správu referencí.

Mezi tyto nástroje patří například:

- EasyBib,
- Bibcitation,
- Paperpile,
- Sciwheel,
- Wizdom.

4.12 Vyhodnocení

Jak již částečně z kapitoly **4.10** plyne, tak ani jeden z nástrojů není vhodným pro použití v této práci. Ačkoli jsou všechny nástroje, až na `RefWorks` zdarma, není u nich buď možnost snadné integrace do již stávajícího systému KIV, nebo nesplňují požadovanou funkcionalitu.

Mezi požadovanou funkcionalitu spadá poloautomatizovaný import dat z vybraného systému a zároveň i automatický export ověřených dat z aplikace přímo do systému KIV. Jednotlivé nástroje umí exportovat publikace v různých výstupních formátech, ale nikoli automatizovaně a není možné tyto záznamy získat pomocí žádné služby, která by navrátila množinu požadovaných dat na základě vstupních parametrů.

[14] `https://www.makeuseof.com/google-docs-add-ons-citation-bibliography/`

5 Návrh řešení

Cílem této práce je vytvoření nástroje pro správu bibliografických záznamů. Tento nástroj musí umět poloautomatizovaný import ze vhodného systému a umožňovat jednotlivé záznamy vhodně prezentovat, například v podobě tabulky. Nástroj musí být integrovatelný do stávajícího systému KIV. Při vývoji musí být brán ohled na možné budoucí rozšíření o další vylepšení. V předchozí kapitole (4.12) byly vyloučeny veškeré nástroje, které byly popsány, jelikož nesplňovaly požadovanou funkcionalitu. Tato kapitola se již zaměřuje na samotný návrh a implementaci aplikace.

Aplikace je velice užitečná zejména pro vědecké pracovníky KIV. Zatím neexistuje žádný nástroj, který by umožňoval snadný poloautomatizovaný vyhledávací mechanismus vědeckých publikací, a zároveň by si ho uživatel mohl přizpůsobovat dle svého uvážení. Následně nabízí několik webových služeb pro integraci zejména se systémem KIV, ale i jiných systémů, díky poskytovaným endpointům několika webových služeb. Slouží tak k urychlení a usnadnění práce.

5.1 Obecný popis aplikace

Jedná se o stand alone[1] webovou aplikaci, jejímž úkolem je starat se o vědecké publikace jednotlivých uživatelů, ať už vědeckých pracovníků KIV, tak i pracovníků z jiných kateder.

Aplikace používá externí API z DBLP databáze bibliografických záznamů. Ta poskytuje vlastní vyhledávací API, které navrátí požadovaný výsledek, ať už je výsledkem myšlen konkrétní autor publikací (jeho jméno atp.), nebo autorovy vydané a zveřejněné publikace s veškerými informacemi ke každé z nich.

DBLP aplikace je jako jediná ze zmíněných bibliografických databází v této práci, která poskytuje veřejné API a zároveň obsahuje citační záznamy publikací vědeckých pracovníků KIV. Z tohoto důvodu je v této práci použita.

Aplikace využívá ke své správné funkcionalitě databázi MySQL, do které ukládá jednotlivé záznamy, které slouží jako cache[2] v případě prvního uložení konkrétního záznamu, který nebyl v minulosti ještě uložen a nebylo

[1] Plnohodnotná samostatná aplikace, která není součástí žádného jiného softwaru

[2] V IT terminilogii se cache používá jako dočasné uložení dat z jiných zdrojů, ke kterým je poté přistupováno jednotně pro urychlení funkcionality.

s ním nijak manipulováno. Tento uložený záznam je poté distribuován skrze jedinečné ID dalším uživatelům, kteří by jej chtěli uložit do své sbírky pod svým konkrétním uživatelským jménem.

Jednotlivé nalezené a uložené záznamy jsou uživatelům poté vykresleny v lidsky čitelné podobě (resp. přehledné tabulce) a manipulace s nimi je prováděna pomocí jednoduchých checkboxů a tlačítek.

5.2 Rozsah projektu

Hlavním cílem projektu je vytvoření webové aplikace včetně jednoduchého uživatelského grafického rozhraní, které umožňuje pohodlně získávat a zpracovávat vědecké články, jenž jsou k dispozici z DBLP databáze. Tyto záznamy si každý uživatel může ukládat a mazat dle svého uvážení.

Uložené záznamy je nutné situovat do dvou množin. Jedna množina představuje uživatelem ručně uložené záznamy s použitím vyhledávacího enginu aplikace a druhá představuje extrahované DBLP záznamy s příznakem správnosti uloženého záznamu, který se váže skrze konkrétní PID[3]. Zde hraje velikou roli webová služba, která exportuje publikace obsahující tento příznak správnosti, díky čemuž je umožněna integrace do stávajícího systému KIV.

Aplikace musí umožňovat co nejpohodlnější a nejrychlejší ovládání pro koncové uživatele. Je proto důležité brát zřetel na co největší automatizaci jednotlivých akcí, které uživatel nemusí dělat, ale aplikace to udělá vše sama za něj.

Nadále je nutné zajištění koncových bodů (`endpointů`), které budou sloužit pro poskytnutí uložených dat z aplikace jiným, externím aplikacím pro následné zpracování záznamů i v jiných systémech. Takový přístup vyžaduje užití několika webových služeb, které by tento problém pohodlně vyřešily jejich zavoláním ať už se vstupními parametry, tak i bez nich. Formát návratových dat by měl být po předchozí domluvě JSON.

5.3 Kontext systému

5.3.1 Třídy uživatelů

Aplikace předpokládá pouze s jednou množinou uživatelů, kteří se budou přihlašovat, případně registrovat při první návštěvě, pomocí vygenerovaného

[3]Person identifier, resp. jedinečné identifikační číslo daného záznamu.

jedinečného odkazu pro každého uživatele zvlášť na univerzitních stránkách katedry KIV.

5.3.2 Provozní prostředí

Aplikace je multiplatformní a napsána v čistém jazyce PHP. Ke své správné činnosti potřebuje server provozující například Apache a předpokládané prostředí, ve kterém aplikace poběží, je server **ares.fav.zcu.cz**.

Aplikace bude integrována se systémem KIV díky implementované webové službě navracející množinu specifických publikací.

5.3.3 Omezení návrhu a implementace

Pro správný běh aplikace je potřeba správné verze PHP, ve které je spuštěná z důvodu zpětné kompatibility jednotlivých verzí. Díky použití objektově orientovaného přístupu a PHPUnit knihovny, která je užita k provozování jednotkových testů kritických částí aplikace, je požadavek na verzi PHP 7.4 a výše pro PHPUnit ve verzi 9.

5.4 Funkce systému

5.4.1 Uživatelské použití - přístup do aplikace

Každý uživatel by se na tuto aplikaci měl dostat skrze vygenerovaný hypertextový odkaz poskytnutý na univerzitní stránce katedry KIV. Odkaz by měl být jedinečný podle přezdívky uživatele (jeho orion účtu). S kombinací orion účtu + jména + příjmení + pevně daného tokenu[4] bude uživatel odkázán do aplikace, která ho automaticky zařadí do jedné ze skupiny:

- uživatel je zde poprvé a jeho přezdívka nebyla v systému zaznamenána → zaregistruje uživatele a automaticky jej přihlásí,
- uživatel je v aplikaci evidován a automaticky jej přihlásí.

5.4.2 Vyhledávací mechanismus aplikace

Nachází se na záložce **Main Page** a je přístupný pouze přihlášeným uživatelům. Do kolonky s názvem *Type author (authors)* uživatel zadá jméno autora (např. P. Veliký), nebo v případě zájmu vyhledání publikací skrze

[4]Token nebo také jinak řečeno pevně daná sekvence, která je akceptovatelná programem a validována pro jedinečnost

více autorů najednou zadá jednotlivá jména autorů oddělena čárkami (P. Veliký, F. Malý, ...). Nadále zadá i maximální počet publikací, jenž bude vypsán pro každého z nich, a to chronologicky v pořadí, v jakém jsou jednotliví autoři zadáni ve vyhledávacím poli. Vzhledem k omezení ze strany DBLP je maximální počet 999 a minimální 1.

Po vyplnění všech potřebných políček (autor, počet) a stisknutí tlačítka *Pull records* se uživateli vykreslí veškeré nalezené záznamy, jenž jsou prezentovány jako tabulky. Ty si může uživatel uložit do své sbírky, která je spojená pouze s jeho uživatelským účtem.

5.4.3 Zajištění správnosti získávaných dat

Po kliknutí na záložku **Account** je uživateli nabídnut seznam jednotlivých autorů včetně odkazů na DBLP (autorů se stejným jménem může být v DBLP evidováno více, proto je důležité, aby se uživatel ujistil a manuálně potvrdil, který DBLP účet je doopravdy jeho). Tento seznam je vygenerován pomocí jména a příjmení evidovaného podle orion účtu v tabulce uživatelů na univerzitních stránkách katedry KIV. Kombinace jména + příjmení je poté zaslána jako textový řetězec do DBLP vyhledávacího API přímo pro autory samostatně, což umožňuje tento elegantní způsob, jak umožnit výběr onoho správného PID.

Po vybrání vhodného autora a zaškrtnutí příslušného checkboxu u konkrétního jména a stisknutí tlačítka *Accept PID* dojde k potvrzení identifikátoru uživatele.

5.4.4 Získání pouze validovaných publikací

Na záložce **My Publications** se zobrazí veškeré publikace vázány pouze ke konkrétnímu uživateli dle jeho orion přezdívky a jeho nastaveného identifikátoru, resp. PID. Je zde seznam publikací evidovaných v DBLP již dle konkrétního PID autora. Při prvním načtení budou záznamy načteny z DBLP aplikace a uloženy automaticky jeden po druhém do databáze, odkud se také budou čerpat pro zobrazení při další návštěvě, kdy se záznamy budou zobrazovat pouze z databáze, nikoli z aplikace DBLP, což urychlý celkový běh aplikace a nebude tak zbytečně zatížena dotazováním se do cizích zdrojů a následnou interpretací.

Zároveň je na této záložce i tlačítko *Refresh*, které slouží pro aktualizování publikací v případě, že by byly nějaké záznamy přidány do DBLP a bylo by tak třeba aktualizovat cache.

5.5 Webové služby poskytované aplikací

Pro zajištění možnosti integrace webové aplikace do jiných existujících systémů je nutné zajištění několika webových služeb poskytujících patřičná data. Webové služby bude možné používat skrze připravené vstupní body, resp. `endpointy` přes dané URI s použitím protokolu HTTP(S).

Pro integraci do stávajících systémů katedry KIV a případně i jiných systémů se nabízí dvě webové služby, přičemž jedna bude sloužit pro export validovaných publikací s příznakem správnosti a druhá by poskytovala DBLP PID každého uživatele registrovaného v aplikaci.

Návratové hodnoty webových služeb konkrétních obsluhovaných požadavků nabývají těchto hodnot:

- **Status: 200 OK:** Poskytnutí veškerých dat,
- **Status: 200 OK:** {"msg": "data not found", "error_code": 1}.

Pozn.: Užití návratové hodnoty 200 v případě nenalezení dat je v tomto případě užito z důvodu, že se vrací alespoň JSON formát s konkrétní chybou, což už samo o sobě návratová data představuje, a koncová aplikace tak bude moct snadno rozlišit, zda dostala množinu dat či chybovou zprávu, a může tak snadno zareagovat dle vlastního uvážení.

5.5.1 Export ověřených publikací

Tato webová služba očekává dva parametry:

- **rok vydání**; v případě nevyplnění se použije aktuální rok,
- **orion přezdívka uživatele**; nepovinný parametr. Pokud je vyplněn, vrátí se všechny publikace uživatele se zadanou orion přezdívkou.

Veškeré publikace jsou s příznakem správnosti, to znamená, že byly v aplikaci uloženy skrze již uživatelem potvrzené PID, které si sám potvrdil.

Přidání speciálního prefix

Je nutné neopomenout přidání prefixu před číslo obd_id, což je unikátní identifikátor publikace evidovaný v systému DBLP. Pro jedinečnost tohoto čísla i ve stávajícím systému KIV bylo nutné přidat předem domluvený prefix **999** před samotné číslo. Tento prefix vyžaduje změnu datového typu sloupečku obd_id z int na bigint. Dojde tak k zamezení problému s přetečením.

5.5.2 DBLP PID všech uživatelů evidovaných v aplikaci

U každého uživatele je od první návštěvy nastavena počáteční hodnota DBLP PID na "none", což znázorňuje, že si uživatel svůj PID nenastavil. Nebo je u uživatele evidována hodnota, kterou si ručně nastavil v aplikaci. Webová služba poté vrátí všechny orion uživatelská jména a jejich hodnotu, kterou mají nastavenou pro PID.

5.5.3 Export všech uložených publikací konkrétního uživatele

Očekává jeden parametr, kterým je **orion** zkratka uživatele, jehož všechny uložené publikace chceme získat.

Oproti webové službě vracející pouze ověřené publikace, tato webová služba vrací všechny publikace, které má konkrétní uživatel uložen ve svém seznamu.

5.6 Použité technologie

5.6.1 PHP

Aplikace byla vyvinuta ve skriptovacím jazyce PHP ve verzi 7.2.10 a plně koresponduje s MVC architekturou.

Tato technologie byla zvolena z důvodu nejednoznačné volby integrace této aplikace do stávajícího systému KIV. Na samém začátku vývoje byla myšlenka integrace v podobě pluginu do současného redakčního systému webových stránek KIV. Pro tento účel byl programovací jazyk PHP vhodný kvůli kompatibilitě aj., jelikož celý redakční systém je v tomto jazyce implementován.

Verze PHP, která je nainstalována na stroji, jenž bude sloužit pro spuštění aplikace, je pevně závislá s verzí, ve které je aplikace vytvořena. Nižší verze PHP by nemusela obsahovat potřebné vlastnosti, které aplikace využívá. A zároveň vyšší verze PHP (nejnovější 8.1) obsahuje změny, které nejsou zpětně kompatibilní s užitou verzí, viz následující kapitola o **PDO** 5.6.3.

Zároveň je aplikace testována i v prostředí běžícím s verzí PHP 7.4, která je aktuálně užívaná na serveru **ares.fav.zcu.cz**, a na kterém je aplikace provozována.

5.6.2 MySQL databáze

MySQL [14] je otevřený systém řízení báze dat uplatňující relační databázový model, jenž je široce využíván ve webových službách, a to zejména díky bezplatné distribuci pod bezplatnou licencí GPL.

Tato technologie byla vybrána z důvodu dostačující funkcionality a to jak z hlediska rychlosti, tak i bezpečnosti. U aplikace není předpoklad, že by byla využívána tisíci uživateli současně v jednu chvíli a hrozilo by tak výrazné zatížení aplikace, které by způsobilo zpomalení aplikace či její nedostupnost. Z pohledu bezpečnosti je také při správném užívání zabezpečena proti útokům zvaným *SQL injection*. Hlavním důvodem je ale použití MySQL databáze v současném systému KIV, kdy jsou oba dva systémy sjednoceny jak technologií jazyka, ve kterém je aplikace vyvinuta, tak i databází, se kterou komunikují.

MySQL s kombinací HTTP(S) Apache serveru a programovacího (resp. skriptovacího) jazyka PHP se může považovat za úplný základ v oblasti vývoje webových aplikací jako takových. Tato kombinace může být součástí například balíčku aplikace XAMPP, která umožňuje zprovoznit lokální server, na kterém je možné provozovat PHP aplikace bez nutnosti třetích stran.

V tomto projektu je použita verze MySQL 15.1 Distrib 10.1.36-MariaDB, která byla dodána současně s verzí PHP 7.2.10 a Apache serveru ve verzi 2.4.34 jako součást balíčku aplikace XAMPP.

5.6.3 PDO

Nebo také **PHP Data Objects Driver** [13] je v aplikaci užit pro bezpečnou komunikaci mezi samotnou aplikací a MySQL (MariaDB) databází.

PDO je ovladač, jenž umí komunikovat až s 13 různými databázemi díky své abstrakci a faktu, že většina databází komunikuje pomocí jazyka SQL, s čímž tento ovladač také počítá. Pří správném užití ovladače je zajištěna ochrana databáze např. před SQL Injection útoky, a jinými.

Velikou výhodou je předpřipravení dotazu do databáze, kdy jsou z aplikace vyžadovány uživatelem zadané vstupní parametry, na základě kterých jsou poté navrácena příslušná data, případně uložena nově vložená data. Tento dotaz je předpřipraven pomocí funkce `prepare()`, kdy jako první vstupní parametr této metody je SQL dotaz, ale jednotlivé parametry jsou např. nahrazeny za znak `"?"`, a jako druhý parametr jsou konkrétní data, jenž budou nahrazovat zástupný znak `"?"` v další fázi.

PDO umí i jiný způsob práce s daty, a to pomocí tzv. bindování, což je také jeden ze způsobů pro manipulaci s jednotlivými volání do databáze,

aniž by byl umožněn útok, díky kterému by útočníkovi byly zobrazena potenciálně důležitá data, která by mohl využít pro své účely.

5.6.4 PHPUnit

PHPUnit framework umožňuje vytváření jednotkových testů, které testují správnou funkcionalitu jednotlivých funkcí uvnitř aplikace, a zajišťuje tak korektní a předvídatelný chod aplikace jako takové.

V aplikaci byl užit zejména pro testování návratových hodnot z DBLP databáze, aby bylo podchyceno co nejvíce případů a zamezilo se tak nečekanému chování aplikace v případě, jako je např. získání nevalidních dat či získání prázdného obsahu. To je zejména důležité v případě, kdy se generuje citace dané publikace, ve které je očekávána přítomnost konkrétních částí textových řetězců, jenž se poté spojují do pevně dané podoby, která tvoří citaci.

Testovaná funkcionalita je oddělena od MVC architektury v samostatném adresáři zvaným "tests", ve kterém se nachází veškeré třídy, jejichž metody jsou testovány. Tato specifikace musí být uvedena v phpunit.xml konfiguračním souboru, ze kterého PHPUnit framework čerpá před každým spuštěním testů příkazem z terminálu .\vendor\bin\phpunit –testdox, přičemž jednotlivé výsledky všech testů definovaných v daném konfiguračním souboru se zobrazí v terminálu, ze kterého se PHPUnit spustil.

5.6.5 HTML5

Pro vykreslení veškerých dat ve webové aplikace je užito značkovacího jazyka HTML, který umožňuje jednoduchým způsobem vizualizování informací do uživatelem čitelné podoby.

U webové stránky je žádané, aby byla členěna na několik základních částí, jinak řečeno fragmenty, ze kterých se celá skládá. Každý fragment má svůj vlastní úkol. Například navigační menu, které má za úkol udržovat navigační okno, nebo patička, která v sobě nese podpis autora.

5.6.6 CSS3, Bootstrap

Použití samotného HTML by bylo pro tuto práci neefektivní vzhledem k již existujícím externím knihovnám a zároveň i CSS[5] jazyku, který definuje, jak jednotlivé HTML elementy budou vypadat.

[5]CSS nebo také jinak řečeno kaskádové styly definující vzhled jednotlivých HTML elementů

Zároveň se nabízí k dispozici i využití svobodné, již zmíněné, knihovny `Bootstrap`, což je otevřená sada nástrojů kaskádových stylů obsahující připravené šablony založené na HTML a CSS. Bootstrap je známý zejména u webových aplikací, jenž jsou implementované i pro jiná zařízení než počítače, kdy je připravena sada několika základních elementů, které lze použít napříč celou aplikací a mohou se jen pomocí krátkých klíčových slov vyvíjet rovnou i na všechna zařízení, která jsou rozdělena do několika skupina (malé, střední, veliké, extra veliké). Toto jsou skupiny, které jsou definované do určité velikosti obrazovky (v pixelech), což znamená, že zařízení, které spadá do konkrétního rozměru uvidí pouze jedno zobrazení elementů pro něj přímo definované.

5.7 Mimofunkční požadavky

5.7.1 Bezpečnost

U každého uživatele musí dojít k autentizaci, aby získal plný přístup k aplikaci. Pokud autentizace neproběhne z jakéhokoli důvodu, uživatel nebude moct manipulovat s aplikací a bude opakovaně vyzýván k přihlášení.

Všechny operace na úrovni modelu, kde je přímý přístup k databázi, jsou plně ošetřeny proti možnému SQL injection útoku.

Jednotlivá data, se kterými je manipulováno, jsou vhodně ošetřena proti XSS útokům (cross site scripting).

Jednotlivé tabulky a data obecně, ke kterým má aplikace přístup a manipuluje s nimi, neobsahují žádné citlivé údaje, které by byly fatální v případě úniku.

5.7.2 Výkon

Výkon závisí na odezvě externí aplikace DBLP, když dochází k přístupu skrze endpointy k datům. Ačkoli není požadavek na data výrazně náročný, může dojít k výrazné latenci (až v jednotkách sekund) ze strany DBLP v případě většího zatížení jejich serveru za předpokladu paralelního přístupu k jejich datům s více cizími aplikacemi naráz.

Tento výkonnostní problém je však řešen pomocí cachování vytažených dat z DBLP do vlastní databáze, což výrazně zrychluje běh celého systému. A to díky tomu, že není znovu zasílán požadavek na získání těch samých dat do DBLP, pokud nedojde ke stisknutí tlačítka *refresh* pro manuální obnovení dat.

5.7.3 Rychlost odezvy

Aplikace má uživatelsky přívětivou odezvu (jednotky milisekund) díky cache paměti a vlastním algoritmům pro přístup k daným datům. Avšak rychlost odezvy je závislá na množství uložených dat a zároveň aktuálnímu počtu přihlášených uživatelů, kteří současně zatěžují aplikaci.

5.7.4 Očekávaný počet uživatelů

Aplikace slouží zejména pro ulehčení práce vědeckých pracovníků, kteří potřebují rychlý přístup k vědeckým článkům, jenž jsou nejlépe aktuální a budou je mít na jednom místě, a to s co nejmenším úsilím z jejich strany. Předpokládané používání se odhaduje na desítky uživatelů.

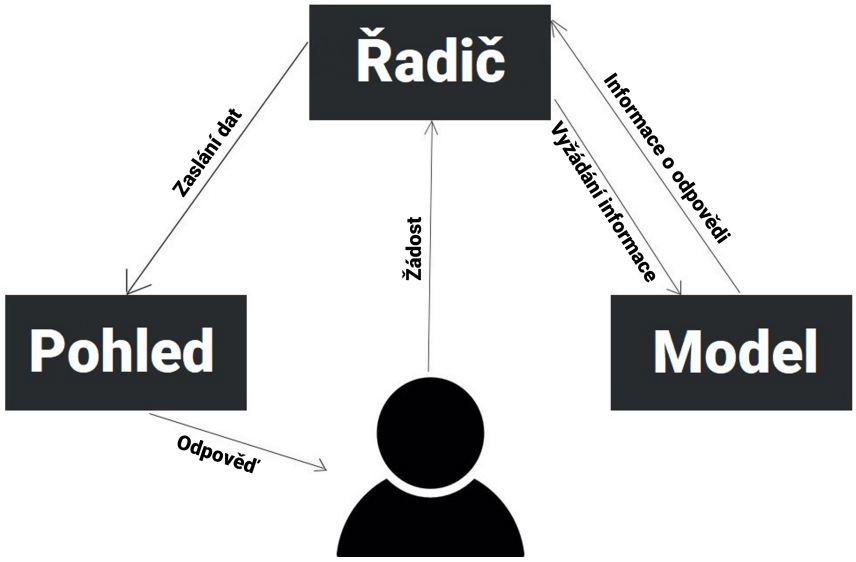
5.7.5 Objem dat

Očekávaný počet záznamů je odhadován na stovky až jednotky tisíců v průběhu užívání aplikace a cachování již někdy v minulosti zobrazených dat z DBLP.

6 Implementace a ověření

6.1 MVC architektura

Aplikace byla vyvíjena dle MVC softwarové architektury, aby bylo možné udržování aplikace a následně byl umožněn i její případný další vývoj do budoucna.



Obrázek 6.1: MVC architektura [16]

Při vývoji aplikace nebylo použito žádného jiného frameworku než PHPUnit. Je to z důvodu kvůli dřívějším požadavkům nad integrací této webové aplikace do redakčního systému, tudíž by to nebyla samostatná aplikace, ale plugin. Tyto požadavky se během vývoje aplikace změnily a integrace do stávajících systémů je realizována skrze webové služby, které aplikace poskytuje. Zároveň je aplikace takto snadno modifikovatelná, což může být další plus při případném budoucím vývoji. Aplikace je rozdělena do několika adresářů:

1. **Kořenový adresář obsahující:**
    - hlavní soubor **`index.php`**: slouží jako hlavní spouštěcí soubor pro celou aplikaci,
    - **`settings.inc.php`**: globální konfigurace celé aplikace. Zde se nachází veškeré důležité údaje používané například pro připojení

k databázi, definice jednotlivých záložek v navigačním menu a a relativní cesty k jednotlivým adresářům. Tyto informace jsou poté používané skrze aplikaci a díky svému umístění jsou i snadno modifikovatelné pouze z jednoho místa,

- adresář **`css`**: obsahuje soubor `style.css`, který v sobě nese veškeré kaskádové styly aplikace. Tyto styly definují vzhled jednotlivým HTLM elementům ve webové aplikaci,
- adresář **`img`**: obsahuje veškeré obrázky použité kdekoli v aplikaci,
- adresář **`tests`**: adresář, který obsahuje jednotkové testy napsané ve Frameworku PHPUnit, kde je testována funkcionalita kritických částí kódu pracujících zejména s DBLP požadavky a kde jsou testovány návratové hodnoty jednotlivých metod,
- soubor **`phpunit.xml`**: konfigurační soubor pro framework `PHPUnit`, ve kterém jsou definovány veškeré údaje zajišťující správnou funkcionalitu frameworku při spouštění jednotkových testů. Jsou zde informace jako je název adresáře, ve kterém se jednotkové testy nacházejí, a zároveň i název testovacího scénáře, který se poté zobrazuje v konzoli nad jednotlivými výsledky.

Veškeré tyto popsané adresáře a soubory uvnitř nich jsou popsány v následující kapitole **6.2**

2. **Adresář app**:

   - Je hlavním adresářem pro celou MVC strukturu. Obsahuje nejen čtyři další podadresáře **Controllers**, **Models**, **Views** a **WebServices**, ale také i třídu `AppStart.class.php` zpracovávající veškeré požadavky zaslané skrze URL.

6.2 Popis tříd

6.2.1 Kořenový adresář: `index.php`

Při spuštění aplikace se zde provedou pouze 3 nejdůležitější operace:

1. načtení globálního nastavení aplikace,
2. načtení třídy (souboru) `AppStart.class.php`,

3. vytvoření instance třídy AppStart a zároveň zavolání implementované metody `AppStart.start()`, kde dochází ke zpracování URL požadavku a podle toho k zavolání správného `"controlleru"` a zároveň i vykreslení "`view`" uživateli.

6.2.2 app/AppStart.class.php

V této třídě je pouze jedna hlavní metoda `start()`, která kontroluje jednotlivé parametry předané skrze URL. V případě, že dané URL obsahuje klíčové slovo:

- **`ws` (zkratka pro webovou službu)**: je buď zavolána funkcionalita dané webové služby v případě dostupnosti (zdroj existuje na daném URI), nebo je uživatel přesměrován na základní, hlavní stránku,
- **`page` (stránka)**: registruje uživatelův požadavek pro získání obsahu na konkrétní záložce, např. `My Saved Data`. V případě, kdy daná vyžadovaná stránka skrze URL existuje, je na ni uživatel přesměrován. Jinak v opačném případě, kdy požadovaná cílová adresa není nalezena, je uživatel automaticky přesměrován na výchozí záložku `Main Page`.

```
1    if (isset($_GET["ws"]) && array_key_exists($_GET["ws"],
   WS_NAMES)) {
2
3        $WS_TYPE = $_GET["ws"];
4        $WS = WS_NAMES[$WS_TYPE];
5
6        require_once(DIR_SERVICES . "/" . $WS["file_name"]);
7
8        /** @var IWebServices $webService */
9        $webService = new $WS["class_name"];
10       $webService->getSpecificData();
11       $webService->returnJson();
12   } else if (isset($_GET["page"]) && array_key_exists($_GET
   ["page"], WPAGES)) {
13       $pageKey = $_GET["page"];
14   } else {
15       $pageKey = DEF_WPAGE;
16   }
17       $pageInfo = WPAGES[$pageKey];
18       require_once(DIR_CONT . "/" . $pageInfo["file_name"])
   ;
19
20       /** @var IController $controller */
21       $controller = new $pageInfo["class_name"];
```

```
22
23          echo $controller->show($pageInfo["title"]);
24      }
```

Listing 6.1: ukázka kódu metody start()

Z ukázky kódu je nutné objasnit, co znamenají klíčová slova `WPAGES` a `WS_NAMES`. Jsou to pole uchovávající klíč ⇒ hodnotu. Přičemž klíč je reprezentován hodnotou z URL a hodnota pod tímto klíčem představuje název konkrétního Controlleru či konkrétní webové služby, která se bude volat.

6.2.3 app/Controller/IController.interface.php

Rozhraní, jenž je implementováno každým Controllerem, kde je definována metoda `show()`, kterou musí mít každá třída vydávající se za Controller. Tato metoda je poté zodpovědná za zavolání konkrétního `view` pro vykreslení veškerého obsahu při dokončení všech odpovědí na uživatelské požadavky.

6.2.4 app/Controller/*Controller.class.php

Zde bude popsán obecný princip jednotlivých controllerů. Jelikož se ve většině z nich dějí z pohledu implementace podobné akce, jako je například reakce na akci, kterou může být nejčastěji stisknutí tlačítka, tak nemá smysl popisování všech metod ve všech controllerech zvlášť.

Veškeré důležité části aplikace a v nich jednotlivé přístupy k datům zde budou popsány, včetně ukázek kódu a vysvětlení.

app/Controller/MainPageController.class.php

Tato třída představující jeden z hlavních controllerů je speciální zejména při reagování na uživatelem zadaná klíčová slova do vyhledávacího mechanismu. Uživatel zde má velice volný přístup při hledání specifických publikací uložených v databázi DBLP aplikace, kdy může využít libovolných klíčových slov pro pohodlné získání všech výsledků. Tyto výsledky se nemusejí týkat přímo konkrétních zaměstnanců na katedře KIV, ale funguje obecně, což může být velice užitečná vlastnost.

Při stisknutí tlačítka *Pull records* dojde k validaci vstupních hodnot uživatelem zadaných do jednotlivých políček. Všechny hodnoty jsou zároveň ošetřeny proti možnému útoku XSS, nebo také jinak řečeno Cross Site Scripting.[1] Ve chvíli, kdy validace proběhne úspěšně, je uživatel přesměrován s již

[1] XSS útok spočívá ve vkládání výkonných částí kódů, například JavaScriptových, které by se mohly poté spustit všem nebo konkrétním uživatelům a způsobit tak škodu.

konkrétními parametry předanými skrze URL adresu na stránku `Searched Publications`, kde se zobrazí veškeré získané výsledky z DBLP endpointu, viz `app/Controllers/OutputController.class.php` níže.

`app/Controllers/OutputController.class.php`

Zde v tomto controlleru dochází již ke spolupráci s modelem `DblpData`, který je zodpovědný za zprostředkování komunikace mezi samotnou aplikací a endpointem poskytnutým z DBLP pro přístup k datům.

Samotný controller pouze požádá model o získání dat s konkrétními parametry, jenž byly zadané uživatelem a zároveň i validované.

```
1  private function validateParametersAndPullDataFromDblp()
2  {
3    global $tplData;
4
5    if (isset($_GET["author"]) && isset($_GET["amount"])) {
6      $author = htmlspecialchars($_GET["author"]);
7      $amount = htmlspecialchars($_GET["amount"]);
8      $tplData["dblp_author"] = $author;
9      $tplData["dblp_amount"] = $amount;
10
11     $this->dblp = new DblpData($author, $amount);
12     $tplData["dblpData"] = $this->dblp->getAllDataFromDBLP();
13   }
14 }
```

Listing 6.2: Ukázka spolupráce s modelem DblpData

Zde v ukázce kódu jsou velice důležité řádky 11 a 12, kdy na 11. řádku dochází k předání konkrétních klíčových slov zadaných uživatelem do modelu `DblpData`. Po instancování modelu s danými parametry dojde uvnitř samotného modelu k přípravě URI, na které se bude s konkrétními vstupními parametry dožadovat návratové hodnoty ve formě XML. Poté se volá na řádce číslo 12 již žádost na model, aby vrátil veškeré výsledky (pokud nějaké získal), a ty se uloží do pole, které je následně předáno dále view pro samotné vykreslení uživateli.

Tento controller nespolupracuje pouze s modelem `DblpData`, ale také i s modelem `Database`, který se stará o veškerou práci s daty už samotné aplikace. V případě, kdy si uživatel zvolí konkrétní publikace, jenž by chtěl uložit a stiskne příslušné tlačítko pro uložení, dojde k zahájení několika činností mezi controllerem a modelem `Database`. Tyto činnosti jsou klíčovými pro celou aplikaci, jelikož zde dochází ke kritické činnosti, a tou je samotná manipulace s daty v databázi, při které nesmí docházet ke vzniku duplicitních záznamů v databázi. Proto je před uložením každé publikace tázáno, zda se

konkrétní záznam již v samotné databázi nenachází. V případě kdy ano, není tento záznam znovu uložen, ale je pouze přiřazen danému uživateli, který si ho chtěl uložit a mohl ho mít ve své sbírce mezi uloženými publikacemi na záložce `My Saved Data`. V jiném případě se záznam vloží do databáze jako nově uložený a zároveň se přiřadí uživateli. Každý záznam může mít přiřazeno více uživatelů najednou přes relaci 1:N.

`app/Controllers/SavedDataController.class.php`

Tento controller reaguje na aktivitu spojenou se záložkou `My Saved Data`. Nedochází zde k ničemu jinému než se samotnou komunikací s modelem databáze, kdy je vyžádáno všech publikací, které si uživatel kdy uložil a má mezi nimi tedy existující relaci v databázi. Tento seznam uložených publikací je poté předán dále příslušnému view, který je všechny zobrazí v přehledné tabulce.

Žádanou funkcionalitou je zde rozšíření o možnost odstranění jednotlivých publikací. Pokud by se tedy uživatel rozhodl, že konkrétní publikaci již nepotřebuje, může jí odstranit jednoduchým kliknutím na tlačítko s ikonou koše. Tento záznam ale nebude odstraněn z databáze, bude pouze odebrána relace mezi uživatelem a samotnou publikací.

`app/Controllers/UserPageController.class.php`

Controller reagující na veškeré uživatelské interakce na záložce `Account`. Spolupracuje jak s modelem databáze, tak i modelem DblData. Vyžádá si jméno a příjmení uživatele od modelu databáze a tuto informaci předá dále modelu DblpData. Tato akce se stane ihned po navštívení stránky `Account`.

Controller zašle modelu databáze orion zkratku konkrétního uživatele pro získání jména a příjmení. Tato informace je poté předána DblpModelu, který zašle požadavek na konkrétní endpoint DBLP aplikace, a to konkrétně pro získání všech autorů s daným či podobným jménem. Pokud je takový uživatel v DBLP evidován, je poslán XML formát s metadaty daného autora, případě všech nalezených autorů najednou. Klíčová informace obsažená v metadatech je PID daného autora.

Pokud uživatel svůj PID nemá stále potvrzený, uvidí na obrazovce množinu všech nalezených autorů z DBLP aplikace. V případě nenalezení žádného autora pro dané jméno a příjmení je vypsána patřičná hláška pro informování uživatele. Pokud je PID již zvolen, je tato informace zobrazena na obrazovce.

app/Controllers/MyPublicationsController.class.php

Klíčový controller, který se stará o dotazování se modelu DblpData na konkrétní publikace již konkrétního autora přes známé PID (pokud ho má uživatel nastavený). V případě první návštěvy záložky `My Publications` je zřejmé, že uživatel žádné vlastní publikace ještě aplikací nedostal. Zavolá se proto příslušná akce z DBLP modelu, kdy je hledáno skrze všechny publikace dle jména a jsou vybrány pouze ty, které splňují shodu s uživatelským PID. Jméno a příjmení aktuálního uživatele je uloženo do databáze při prvním navštívení stránky přes vygenerovaný hypertextový odkaz, a toto jméno a příjmení je zasláno do DBLP při získání příslušných dat.

Veškeré získané záznamy jsou následně ukládány do databáze a je jim nastaven příznak správnosti, který slouží jako identifikátor pro webovou službu, která tyto záznamy poté poskytuje dále. Zároveň platí, že pokud daný záznam v databázi již existuje, je mu pouze změněna hodnota příznaku správnosti, ale nedochází k vytvoření duplicity v databázi.

6.2.5 app/Models/*

Model obecně slouží pro zprostředkování dat, které žádá jakýkoli controller, a v této aplikaci tomu není jinak. Proto tato práce stručně popisuje zejména `DblpData` a `Database` modely, přičemž z DblpData modelu jsou popsány důležité konstrukce a u databázového jsou popsány obecně, jelikož neslouží k ničemu jinému, než k volání SQL dotazů do MySQL databáze pro manipulaci s daty.

app/Models/DblpData.class.php

Velice důležitý model, který je jádrem celé aplikace, zprostředkovávající komunikaci mezi DBLP databází a samotnou aplikací. Dochází zde ke generování specifických URI na základě konkrétních parametrů, které mohou být zadány buď jako uživatelské vstupy, nebo vygenerované aplikací pro získání potřebných dat. Na vygenerovaném URI je poté odeslána žádost DBLP aplikaci o získání potřebných dat, mezi kterými je následně iterováno a jsou ukládány do asociativního pole, které je předáno konkrétnímu controlleru, jenž si o data žádal.

```
1     private function getXMLFromDBLP($url): string
2     {
3         return file_get_contents($url);
4     }
```

Listing 6.3: Ukázka získání dat z DBLP

Zde na ukázce kódu z modelu je vidět funkce, které je předán parametr URL, na který pošle žádost a navrátí získaný obsah (pokud nějaký je) ve formě textového řetězce. Ten je následně zpracován knihovní funkcí jazyka PHP, resp. `SimpleXMLElement`, který zpracuje XML obsah a vytvoří z něj datovou strukturu, nad kterou je možné iteračně získávat hodnotu za hodnotou například následujícím způsobem:

```
1  private function iterateThroughXMLWithAuthors(
       SimpleXMLElement $parseXML) {
2
3   for ($i = 0; $i < count($parseXML->hits->hit); $i++) {
4    $this->authorsArray[] = array(
5      "author" => $parseXML->hits->hit[$i]->info->author,
6      "url" => $parseXML->hits->hit[$i]->info->url,
7      "checkboxValue" => self::$indexCounter
8    );
9
10   self::$indexCounter++;
11  }
12 }
```

Listing 6.4: Ukázka iterace nad zpracovaným XML

app/Models/Database.class.php

Druhý nejdůležitější model v celé aplikaci, který řeší komunikaci s MySQL databází s použitím ovladače PDO.

Ovladač PDO umožňuje spojení až s 13 různými databázemi a řeší veliké množství kritických činností, na které by mohl běžný programátor zapomenout. Při správném použití je důležitý zejména díky ochraně před SQL injection útokem a to hned několika různými způsoby. Prvním způsobem je mapování jednotlivých vstupních parametrů pomocí klíčového slova *bindParam* nebo bindValue, a druhým způsobem, a zároveň v aplikaci použitým, je příprava samotného SQL dotazu se znakem ? místo konkrétní hodnoty, se kterou je dotaz zavolán. Hodnoty jsou dodány jako sekundární parametr zaslány samotné funkci, která řeší komunikaci s MySQL databází.

```php
public function addNewUser($nickname, $fname, $lname): bool
{
    $q = "INSERT INTO
            app_users (`nickname`, `fname`, `lname`)
            VALUES (?, ?, ?)";

    $params = [$nickname, $fname, $lname];
    return $this->query($q, $params);
}
```

Listing 6.5: Ukázka zavolání dat pomocí PDO

Celý tento model se skládá z podobných funkcí, které slouží zejména pro zavolání potřebných SQL dotazů skrze PDO do MySQL databáze pro získání, vložení, odstranění či modifikaci konkrétních dat, přičemž je dbáno kvality při vytváření konkrétních dotazů tak, aby byly co nejefektivnější, tzn. s co největším využitím samotného SQL jazyka a co nejméně PHP kódu.

6.2.6 app/Views/*

Zde se nachází zejména šablony, nebo také jinak řečeno jednotlivé views, které již přebírají od controllerů, jenž je zavolají, potřebná data a ta vykreslí uživateli v definované podobě.

Je vytvořena základní třída `TemplateBasics.class.php`, ve které jsou definované jednotlivé elementy, jenž jsou poté vykreslovány napříč konkrétními views. Jednotlivé metody obsahují HTML elementy tvořící daný obsah.

Názorná ukázka na view pro záložku `Main Page`:

```php
<?php
global $tplData;
require_once(DIR_VIEWS . "/TemplateBasics.class.php");
$template = new TemplateBasics();
$template->getHeader($tplData['title']);
$template->getBreadcrumb("...");

if (isset($_SESSION["userKey"])) {
    $template->mainPageContent();
} else {
    $msg = "<p style='text-align: center;'>Unfortunatelly,
    this section is for logged users only. To continue, please
     <a href='index.php?page=user'>log in</a></p>";
    $template->err->error_msg_warning($msg); // Not Logged In
     err_msg
}
$template->getFooter();
```

Listing 6.6: Ukázka view

Každý view zobrazující se uživateli se skládá z postupně volajících fragmentů napříč všemi view stejně jako je vidět v ukázkovém kódu. Je tím docíleno neopakujícím se částem jednotlivých fragmentů v případě, kdy by se jedna část volala napříč více view, běžně také například u patičky nebo hlavičky, ve které se nachází aplikační menu.

```
1
2 public function getBreadcrumb($path1 = "")
3 {
4  ?>
5   <nav aria-label="breadcrumb">
6      <ol class="breadcrumb bg-white">
7        <li class="breadcrumb-item"><a href="index.php" style=
     "color: #e0b100; font-weight: bold;">Main Page</a></li>
8      <?php
9      if ($path1 != "") {
10     ?>
11         <li class="breadcrumb-item active aria-current='page'
     "><?= $path1 ?></li>
12     <?php
13         }
14     ?>
15       </ol>
16   </nav>
17   <?php
18 }
```

Listing 6.7: Ukázka části z TemplateBasics.class.php

Zde je vidět ukázka kódu z `TemplateBasics.class.php`, kde je vytvořena metoda uvnitř které je definováno menu *breadcrumb*, což je speciální druh menu ukazující aktuální úroveň, ve které se uživatel na stránce nachází, a které mu také umožňuje snadnější navrácení o nějakou úroveň zpět. Toto menu poté může vypadat následovně:

Main Page / Current Page

což indikuje, že uživatel otevřel záložku Current Page z předchozí stránky Main Page. Tudíž je zde jakési trasování záložek, ve kterých se nachází, a které má stále před sebou a může jedním kliknutím jít o úroveň zpět.

6.3 Webové služby poskytované aplikací

6.3.1 Získání publikací uložených v aplikaci

Webová služba, kterou lze zavolat několika způsoby:

1. bezparametrově:
   - **URI**: ⟨adresa⟩`/index.php?ws=dblp_publications`
   - Automaticky se vrátí publikace **všech autorů** pro **aktuální** rok.
2. s parametrem year:
   - **URI**: ⟨adresa⟩`/index.php?ws=dblp_publications&year=`
   - Očekávána číselná hodnota o délce 4, např. 2022
3. s parametrem name:
   - **URI**: ⟨adresa⟩`/index.php?ws=dblp_publications&name=`
   - Očekávaný parametr je jméno a příjmení, např. Petr Novák
4. **kombinace parametrů**: `&name=`⟨hodnota⟩`&year=`⟨hodnota⟩

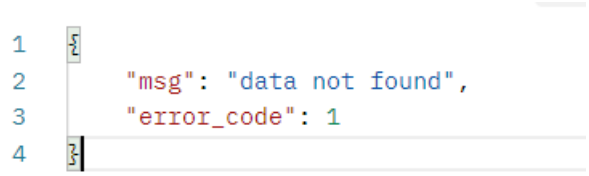
Data jsou navráceny ve formátu `JSON` (ukázka):


```
{
    "obd_id": 99923672,
    "dblp_key": "journals/nca/MartinekLK22",
    "nazev": "Correction to - Building an efficient OCR system for historical d
    "nazev_orig": "Correction to - Building an efficient OCR system for histori
    "nazev_en": "Correction to - Building an efficient OCR system for historica
    "rok": 2022,
    "citace": "Jirí Martínek, Ladislav Lenc, Pavel Král: Correction to - Buildi
        , 2022",
    "doi": "10.1007/S00521-020-05563-6",
    "forma_obd": "Článek"
},
```


Obrázek 6.2: Ukázka s parametrem **name=Pavel Král**

Z důvodu velikosti (šířky) řádku návratových hodnot nebylo možné vložit obrázek s celou šířkou. Byl proto oříznut, ale na ilustraci návratových dat to nic nemění.

V případě nenalezení žádných údajů je navrácena příslušná hláška, na kterou může systém KIV adekvátně zareagovat:



Obrázek 6.3: Ukázka chybové hlášky při nenalezení žádných dat

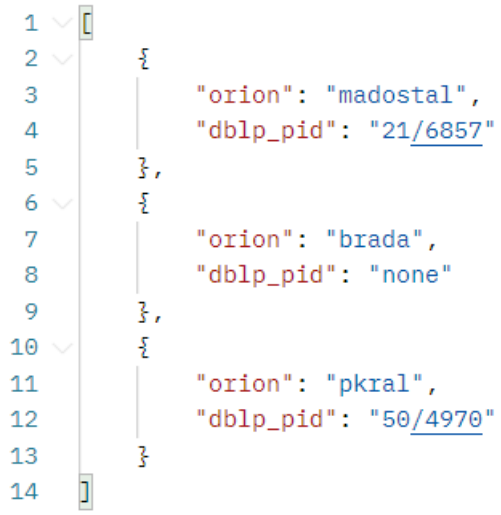
6.3.2 Získání PID všech uživatelů registrovaných v aplikaci

Webová služba je bezparametrová a má za úkol navrátit všechny evidované uživatele v aplikaci a jejich konkrétní PID získaný přímo z DBLP systému.

Webová služba se nachází na následujícím URI:

⟨adresa⟩/index.php?ws=dblp_get_pids

Návratová hodnota je taktéž ve formátu JSON a vypadá následovně:



Obrázek 6.4: Návratová hodnota webové služby pro získání všech PID uživatelů

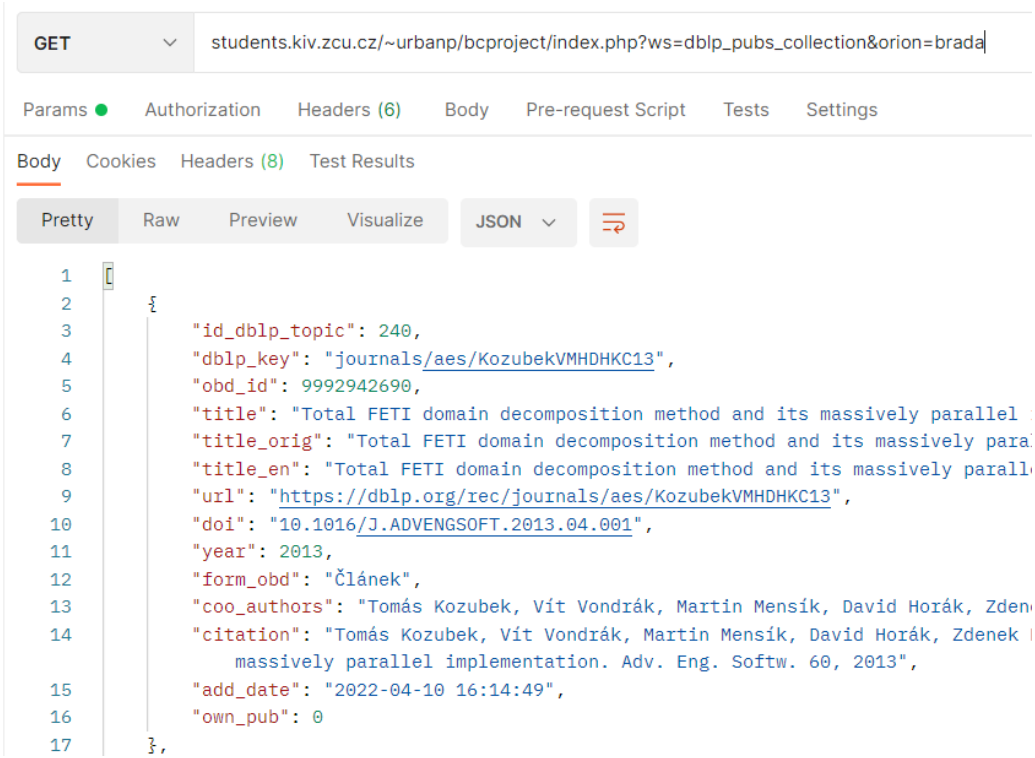
V případě nenalezení žádných hodnot je navrácena znovu chybová hláška, viz obrázek **6.3**

6.3.3 Získání všech uložených publikací konkrétního uživatele

Webová služba, která má za úkol vrátit seznam všech uložených publikací konkrétního uživatele. Vyžaduje pouze jeden vstupní parametr, kterým je orion zkratka uživatele. Webová služba se nachází na následujícím URI:

⟨adresa⟩/index.php?ws=dblp_pubs_collection&orion=

Návratová hodnota je jako u předchozích webových služeb ve formátu JSON a vypadá následovně:



Obrázek 6.5: Získání všech uložených publikací konkrétního uživatele

V případě nenalezení žádných hodnot je navrácena znovu chybová hláška, viz obrázek **6.3**

6.4 Případy užití

V této kapitole budou rozepsány případy užití celé webové aplikace. Pro ilustraci u jednotlivých bodů jsou přiloženy obrázky znázorňující aktuální případ.

6.4.1 Přístup do webové aplikace

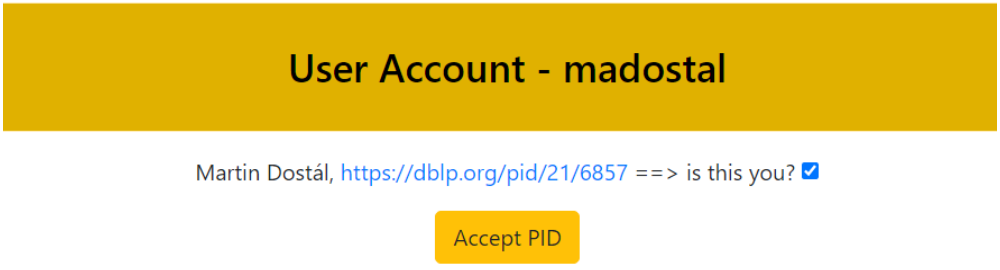
Přístup do webové aplikace je zřízen přes automaticky vygenerovaný hypertextový odkaz umístěný na stávajících webových stránkách katedry KIV. Každý zaměstnanec by měl mít pro svůj orion účet vlastní odkaz, který obsahuje všechny potřebné parametry pro správné navázání spojení s webovou aplikací. Hypertextový odkaz se skládá z několika parametrů:

1. `action=login`: parametr, díky kterému aplikace bude očekávat akci pro přihlášení / novou registraci,
2. `orion=`: parametr, kterým je orion název účtu konkrétního uživatele, např. urbanp,
3. `firstname=`: jméno uživatele,
4. `surname=`: příjmení uživatele,
5. `token=`: pevně daný token, který byl specifikován v požadavcích této práce. Lze ho kdykoli změnit z konfiguračního souboru `settings.inc.php`.

Při první návštěvě je uživatel registrován v systému a zároveň dojde i k jeho přihlášení, tudíž může okamžitě začít využívat webovou aplikaci. Pokud již na webové aplikaci má zřízen účet, bude automaticky přihlášen.

6.4.2 Nastavení DBLP PID

Je nutné, aby si každý uživatel zvolil své PID, pod kterým je veden v DBLP systému. Díky tomuto PID bude zprovozněna funkcionalita na záložce `My Publications`, kde se automaticky dotahují veškeré autorovy publikace vázané právě skrze tento PID. Publikace stažené tímto způsobem jsou zároveň exportovány ven skrze webovou službu.



Obrázek 6.6: zvolení DBLP PID

Toto je jediná manuální činnost, která je v aplikaci vyžadována, jelikož by bylo obtížné najít jiný způsob, kterým by to šlo ověřit. Pokud se uživateli na záložce Accounts v této sekci objeví tímto způsobem více uživatelů pod stejným či podobným jménem, je nutné, aby prošel veškeré odkazy a vybral si doopravdy ten, jehož účet na DBLP mu náleží a potvrdil tuto volbu zaškrtnutím *checkboxu*, viz obrázek **6.6**, a stisknutím tlačítka Accept PID pro potvrzení volby.

6.4.3 Stáhnutí všech vlastních publikací

Tato činnost je dostupná na záložce `My Publications` ve chvíli, kdy je splněna kapitola **6.4.2**, a je zcela automatizovaná. Při prvním kliknutím dojde k automatickému dotažení všech záznamů, které jsou následovně uloženy jeden po druhém do databáze v případě, kdy tam ještě daný záznam uložen nebyl, nebo je u něj pouze aktualizován příznak správnosti indikující webové službě, že může být exportován do stávajícího systému KIV s použitím webové služby nabízenou aplikací.

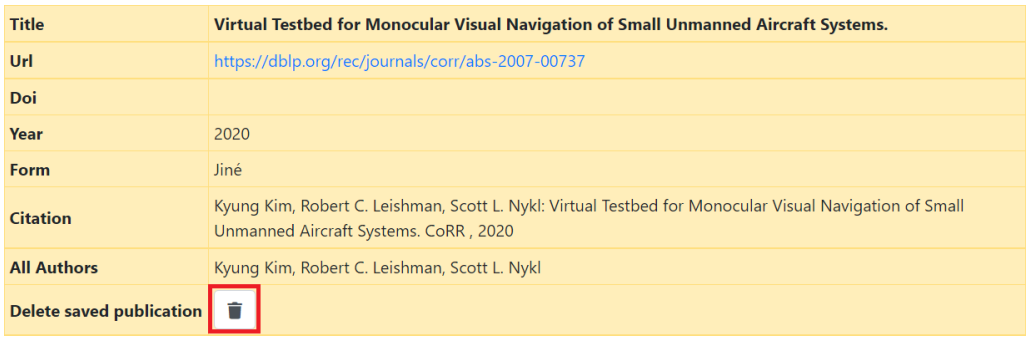
Obrázek 6.7: Aktualizace publikací

Ve chvíli, kdy se uživateli automaticky stáhnou všechny jeho publikace, má k dispozici tlačítko `Update my publications from DBLP`, viz obrázek **6.7**, které zajistí aktualizování všech záznamů.

6.4.4 Zobrazení všech uložených publikací

Na záložce `My Saved Data` se nachází všechny publikace, které uživatel **uložil sám** s použitím vyhledávacího mechanismu a po stisknutí tlačítka `Save Records`, viz kapitola **6.4.5**. Tyto publikace, které si uživatel uložil nemají nastaven příznak správnosti. Protože si uživatel může najít jakoukoli publikaci chce a uložit si publikace i zahraničních autorů, není proto požadováno, aby tyto záznamy byly exportovány webovou službou.

Jsou zde tedy zobrazeny publikace, které si uživatel uložil sám, a které si může také libovolně mazat ze svého seznamu stisknutím příslušného tlačítka s ikonou koše.



Obrázek 6.8: Smazání záznamu

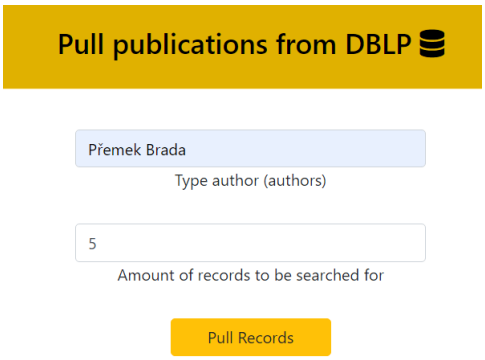
6.4.5 Vyhledání libovolných publikací

Základní funkcionalita webové aplikace, u které uživatel nemusí mít nastaven ani svůj DBLP PID.

Do první kolonky uživatel zadá autora či více autorů, jejichž publikace chce najít ⟨`jméno příjmení`, `jméno příjmení`⟩; v případě více autorů je nutné je oddělovat čárkou ",". Do druhé kolonky zadá celkové množství publikací, které pro daného autora či autory chce získat. Je možné pro konkrétního autora vyhledat pouze např. jeho tři publikace, přičemž není zaručeno, že jsou to ty nejnovější, je to pouze omezení na počet získaných záznamů. Pokud uživatel tedy chce vyhledat **všechny** publikace jednoho konkrétního autora, je vhodné zadat minimální počet dostatečně veliký tak, aby pokryl všechny možné publikace. Avšak DBLP má sám omezené množství záznamů, které navrací, tudíž je aplikace na tuto situaci již připravena a minimální a maximální počet záznamů automaticky hlídá. Konkrétní hodnota se může pohybovat mezi $\langle 1, 999 \rangle$, přičemž doporučená hodnota je například 100, protože pokud autor vydal například 20 publikací a uživatel jich chce 100, vrátí se mu pouze těch 20 vydaných.

Po vyplnění všech patřičných políček a stisknutí tlačítka `Pull Records` dojde k navrácení všech nalezených záznamů z DBLP. Každý záznam je zobrazen v lidsky čitelné podobě, a tou je tabulka:

Pokud by se uživatel rozhodl, že si chce konkrétní publikaci, případně více publikací, uložit, stačí je všechny označit stisknutím "checkboxu" u konkrétní položky, viz obrázek **6.10** a poté stisknout tlačítko `Save chosen publications`, které se nachází dole na konci stránky, viz obrázek níže **6.11**.



Obrázek 6.9: Vyhledání publikací

Author	**Přemek Brada**
Title	The Empirical Evaluation of Semantic Alignment Quality Metrics for Vehicle Domain Component Frameworks Interface Ontologies.
Url	https://dblp.org/rec/conf/flairs/DeBMN21
Doi	10.32473/FLAIRS.V34I1.128512
Year	2021
Key	conf/flairs/DeBMN21
Venue	FLAIRS Conference
Type	Conference and Workshop Papers
Citation	Sangita De, Premek Brada, Jürgen Mottok, Michael Niklas: The Empirical Evaluation of Semantic Alignment Quality Metrics for Vehicle Domain Component Frameworks Interface Ontologies. FLAIRS Conference , 2021
All-Authors	Sangita De, Premek Brada, Jürgen Mottok, Michael Niklas
Save Publication	

Obrázek 6.10: Nalezená publikace

Po stisknutí tlačítka se veškeré zvolené publikace uloží a uživatel je může najít na záložce `My Saved Publications`.

6.5 Testování aplikace

6.5.1 PHPUnit testy

Pro kritické části kódu vykonávající důležitou část při manipulaci s daty získaných z DBLP jsou napsané jednotkové testy pomocí PHPUnit, aby bylo ověřeno, že jednotlivé metody vracejí správné hodnoty a adekvátně reagují na správné i špatné vstupy.

Jednotkové testy byly napsány zejména nad třídou `DblpData.class.php`, kde dochází ke zpracování dat ve formě XML a mapování typů publikací. Bylo otestováno chování metod, které například zpracovávají přijatý XML

Obrázek 6.11: uložit publikace

obsah, kde byla nasimulována některá z chybějících hodnot.

Z důvodu dynamicky měnících se dat v čase nebylo možné napsat jednotkové testy pro metody pracující přímo se samotnými endpointy DBLP aplikace. Jednotkové testy by měly být napsány tak, aby byla vstupní množina testovacích dat stejná, a to při každém spuštění testu. Lze poté porovnávat očekávaný výstup s aktuálním výstupem z dané metody a zjistit tak případnou chybu.

6.5.2 Funkcionální testování

Tento způsob testování byl aplikován při jakékoli nové změně implementace webové aplikace. Bylo simulováno několika uživatelských scénářů, které měly za úkol ověřit správné chování každé komponenty v aplikaci. Tyto testovací scénáře byly vytvořeny na základě případů užití ze strany uživatelů, jenž plynou ze specifikace požadavků, viz kapitola **5**, a byly prováděny vlastnoručně.

Do všech vstupních políček určených pro interakci s uživatelem byly vkládány vhodné množiny dat, které měly za úkol simulovat správnou reakci aplikace na chybové vstupy a zároveň i správné chování aplikace při korektních vstupních hodnotách.

6.5.3 Testování webových služeb

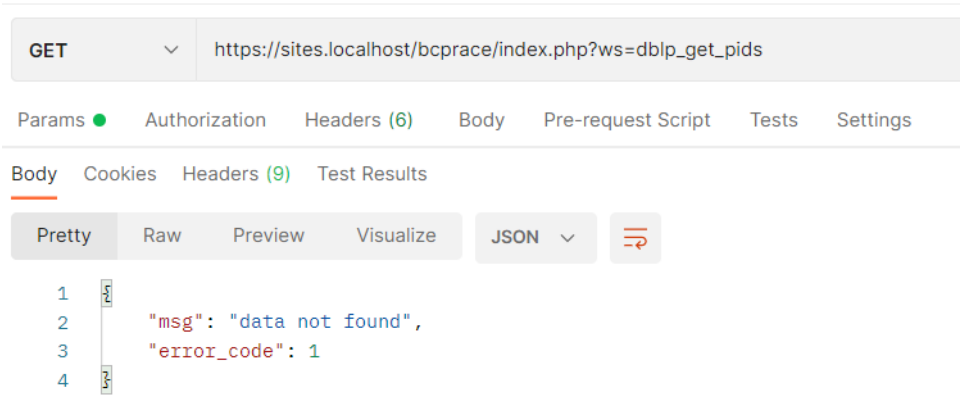
Webové služby byly důkladně testovány pomocí nástroje POSTMAN. Tento nástroj vyžaduje pro svoji funkcionalitu zvolení příslušného HTTP požadavku (`GET`, `POST`, `PUT`, `DELETE`) a URI, ze které se zavolá specifická webová služba. Obě tyto služby mají návratovou hodnotu ve formátu JSON.

První webová služba

První webová služba má za úkol vracet seznam všech registrovaných uživatelů v aplikaci a PID, který je u každého přiřazen. Pokud si uživatel doposud svůj PID stále nezvolil, je místo něj dosazena hodnota “none“. Pokud

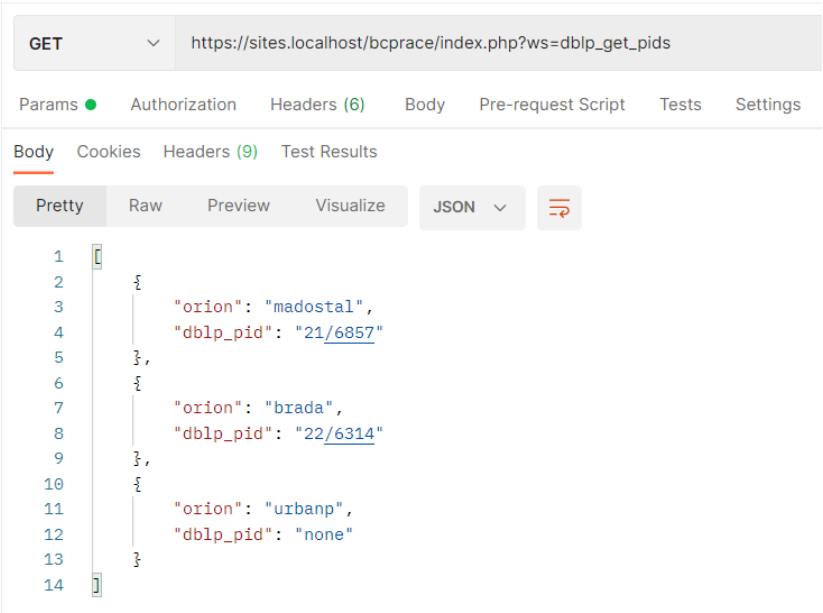
by nastala situace, kdy by nebyly navráceny žádná data, je očekáván výstup s chybovou zprávou “data not found“ a chybovým kódem 1.

Nejdříve je otestována situace, při které v systému není evidován žádný uživatel a webová služba na to vhodně zareaguje, viz obrázek **6.12** níže.



Obrázek 6.12: Chybová hodnota první webové služby

Následně byla otestována běžná situace, při které je v systému alespoň 1 uživatel a webová služba tedy má data k navrácení, viz obrázek **6.13** níže.



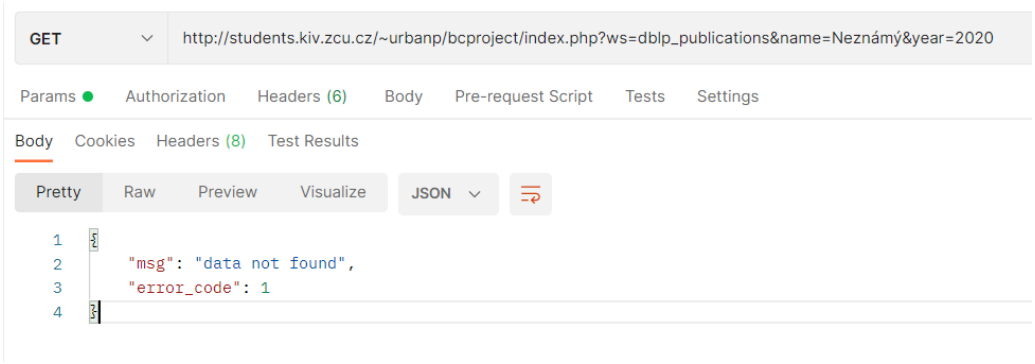
Obrázek 6.13: Navráceny data s uživateli a jejich PID hodnotami.

Druhá webová služba

Tato webová služba je klíčovou pro celou aplikaci, jelikož je díky ní umožněna integrace do stávajících systémů katedry KIV.

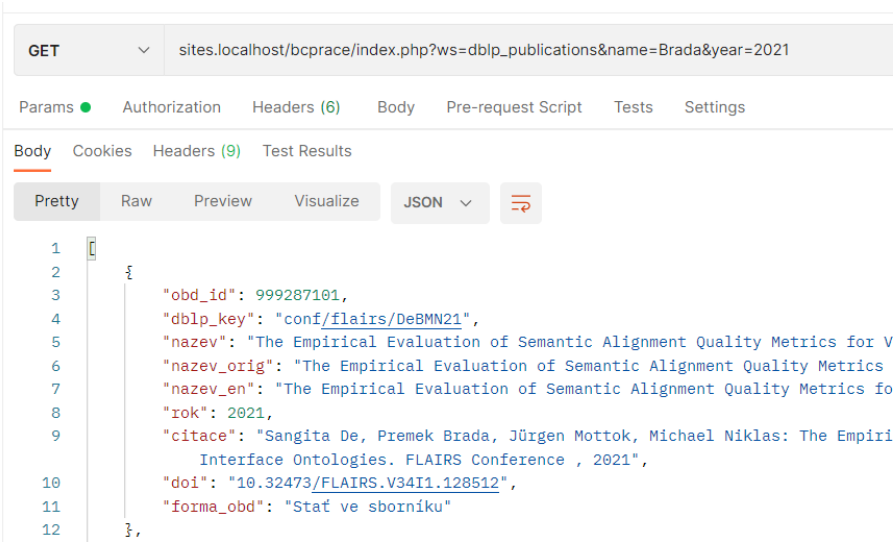
Jako první bylo otestováno chování při nenalezení žádných dat pro konkrétní parametry, které do webové služby vstupují. Tato situace může nastat například ve chvíli, kdy se v databázi nenachází žádné publikace připravené pro export. Nebo žádná z publikací neobsahuje hledaná klíčová slova.

Níže je uveden obrázek **6.14** s výstupem testu, ve kterém je nasimulována situace pro nenalezení žádných dat. Tudíž je navrácena chybová hláška s chybovým kódem.



Obrázek 6.14: Navrácena chybová zpráva z webové služby v případě nenalezení žádné publikace

Zároveň je zde přiložen i obrázek po vyhledání publikací obsahující hledaná klíčová slova. Je nutno znovu podotknout, že publikace vystupující z webové služby jsou již validované tzv. příznakem správnosti. Ten je automaticky nastaven ve chvíli, kdy uživatel potvrdí svůj PID a klikne na stránku `My Publications`. Ukázku navrácených dat lze vidět níže na obrázku **6.15**.

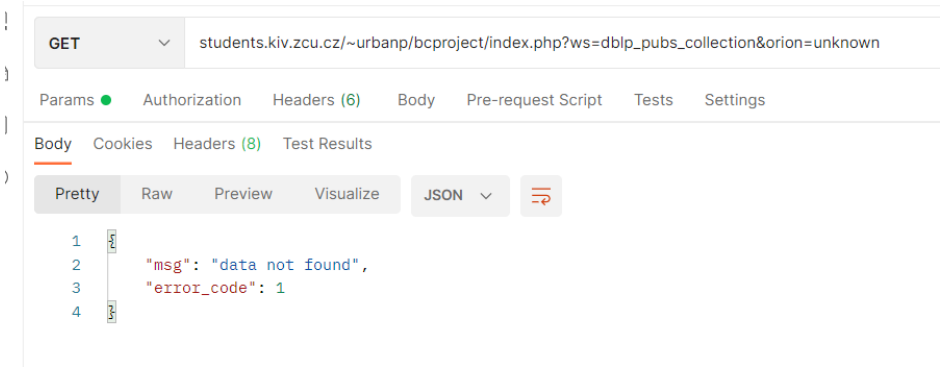


Obrázek 6.15: Navráceny publikace obsahující hledaná klíčová slova

Třetí webová služba

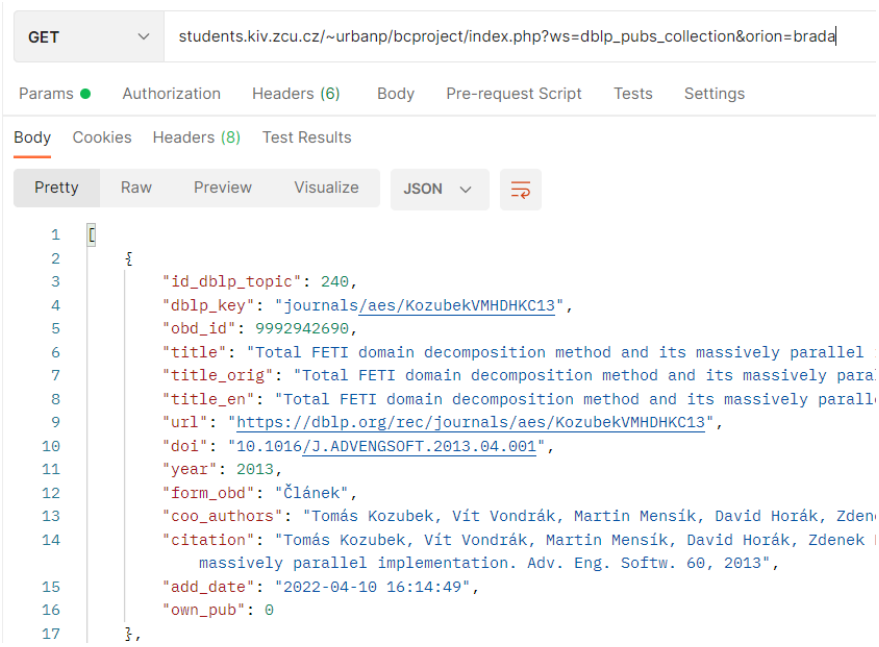
Webová služba, která je rozšířením první webové služby **6.5.3**. Po jejím zavolání s parametrem orion zkratky konkrétního uživatele budou navráceny všechny publikace, které má konkrétní uživatel uložené ve svém seznamu.

V případě nenalezení žádných dat (uživatel nemá žádné uložené) bude navrácen patřičný JSON s patřičnou chybovou hláškou.



Obrázek 6.16: Navrácená chybová zpráva z webové služby v případě nenalezení žádné publikace

V jiném případě, kdy konkrétní uživatel má uložené některé publikace ve svém seznamu, bude navrácena celá kolekce těchto publikací.



Obrázek 6.17: Navrácení všech uložených publikací daného uživatele

6.5.4 Testovací skript

Pro ověření správné integrace dat do stávajícího systému KIV byl napsán jednoduchý testovací skript, který má za úkol ověření zda:

1. webovou službu lze zavolat z kódu s konkrétními parametry,
2. jsou navrácena správná data,
3. je data možné převést z formátu JSON do asociativního pole,
4. se správně rozpozná, jestli byla navrácena data nebo chybový kód,
5. se správně vytvoří SQL příkaz v případě, že nedošlo k odchycení chybové hlášky,
6. dojde ke vložení všech získaných publikací do kopie originální tabulky systému KIV uchovávající všechny publikace.

Po spuštění skriptu bylo dosaženo správného (očekávaného) chování celé aplikace a integrace byla úspěšná.

7 Závěr

Prvním krokem této práce je studie webových služeb a technologií s nimi spojovaných. Na základě této studie webových služeb bylo následně postupováno při výběru vhodných webových technologií pro tvorbu aplikace. Následně je v práci analyzována funkcionalita několika bibliografických databází a na základě této analýzy byla vybrána ta nejvhodnější, kterou je DBLP. Tato bibliografická databáze jako jedna z mála obsahuje veliké množství uložených publikací vědeckých pracovníků KIV, a zároveň i volně přístupnou webovou službu.

Na základě provedených studií a analýz byla navržena aplikace tak, aby umožňovala poloautomatizovaný import publikací z DBLP zasláním požadavku na endopint webové služby poskytovanou DBLP aplikací. Tyto data jsou získána v podobě XML, která jsou následovně zpracována v aplikaci a vhodným způsobem jsou zobrazena ve formě tabulky. Každý uživatel si poté může jakýkoli záznam uložit, případně již uložený záznam může odstranit.

V průběhu implementace byl brán zřetel na nejdůležitější funkcionalitu aplikace, kterou je integrace do stávajících webových stránek KIV. Integrace webové aplikace byla umožněna poloautomatickým získáním ověřených publikací konkrétních uživatelů evidovaných ve webové aplikaci skrze ověřené PID. Tyto záznamy jsou uložené v databázi s nastaveným příznakem správnosti. Publikace s tímto příznakem jsou poté exportovány po zavolání příslušné webové služby, která bude volána v pravidelných intervalech ze stávajícího systému KIV.

Integrace webové aplikace do stávajícího systému se díky implementování webové služby povedla. Aplikace může být také integrovatelná do libovolného systému, nikoli pouze systému KIV.

Aplikace byla průběžně důkladně otestována jak funkcionálním testováním, tak i jednotkovými testy pro kvalitní otestování samotné implementace kritických částí aplikace. Největší důraz byl kladen na samotné testování integrace webové aplikace, jenž byla otestována pomocí automatizovaného skriptu.

Do budoucna by aplikace mohla být rozšířena přidáním dalšího zdroje, který by sloužil pro získávání jednotlivých publikací současně se zdrojem DBLP. Získaná data by se dala spojit a bylo by tak možné získat více informací, než jaké je aktuálně získáváno z DBLP.

Literatura

[1] A. Gokhale, A. S. B. K. *Reinventing the Wheel? CORBA vs. Web Services* [online]. ra.ethz.ch. [cit. 2022/04/22]. Dostupné z: https://www.ra.ethz.ch/cdstore/www2002/alternate/395/index.html.

[2] Berners-Lee, M. . M. *Uniform Resource Locators (URL)* [online]. datatracker, 1994. [cit. 2022/04/21]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc1738.

[3] Cunningham, J. M. *Bibliography* [online]. Britannica. [cit. 2021/12/10]. Dostupné z: https://www.britannica.com/topic/bibliography.

[4] Fitzpatrick, J. *Cite: A WordPress Plugin to Help Readers Correctly Attribute Articles* [online]. WPTAVERN, 17/08/2009. [cit. 2021/12/19]. Dostupné z: https://lifehacker.com/mendeley-manages-your-documents-on-your-desktop-and-in-5334254.

[5] *Uniform Resource Locators (URL)* [online]. tutorialspoint, 2022. [cit. 2022/04/21]. Dostupné z: https://www.tutorialspoint.com/http/http_requests.htm.

[6] Gooding, S. *Cite: A WordPress Plugin to Help Readers Correctly Attribute Articles* [online]. WPTAVERN, 15/04/2014. [cit. 2021/12/19]. Dostupné z: https://wptavern.com/cite-a-wordpress-plugin-to-help-readers-correctly-attribute-articles.

[7] Hendrix, I. C. RefWorks. *Journal of the Medical Library Association.* January 2004, 92, 1, s. 111–3. ISSN 1558-9439 (Electronic). Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC314118/.

[8] *HTTP: A protocol for networked information* [online]. W3. [cit. 2022/02/19]. Dostupné z: https://www.w3.org/Protocols/HTTP/HTTP2.html.

[9] *JSON* [online]. json.org. [cit. 2022/02/20]. Dostupné z: https://www.json.org/json-en.html.

[10] Linthicum, D. *Service Oriented Architecture (SOA)* [online]. Microsoft. [cit. 2022/04/22]. Dostupné z: https://web.archive.org/web/20170707052149/https://msdn.microsoft.com/en-us/library/bb833022.aspx.

[11] Malý, M. *REST: architektura pro webové API* [online]. zdrojak.cz, 2009. [cit. 2022/03/06]. Dostupné z: https://zdrojak.cz/clanky/rest-architektura-pro-webove-api.

[12] Nilo Mitra, W. E. Y. L. *SOAP* [online]. w3, 2007. [cit. 2022/03/05]. Dostupné z: https://www.w3.org/TR/soap12-part1/.

[13] *(The only proper) PDO tutorial* [online]. phpdelusions.net. [cit. 2022/03/20]. Dostupné z: https://phpdelusions.net/pdo.

[14] Richard, B. *What is MySQL: MySQL Explained For Beginners* [online]. hostinger.com, 2017. [cit. 2022/03/20]. Dostupné z: https://www.hostinger.com/tutorials/what-is-mysql.

[15] *Difference Between RPC and Web Service (With Table) – Ask Any Difference* [online]. Ask Any Difference, 2019. [cit. 2022/23/01]. Dostupné z: https://askanydifference.com/difference-between-rpc-and-web-service/.

[16] Selvidge, N. *MVC* [online]. natasha-selvidge, 2021. [cit. 2022/03/05]. Dostupné z: https://natasha-selvidge-98993.medium.com/mvc-9bbfdf3ab6c6.

[17] Valeš, Z. Určování nahraditelnosti a kompatibility webových služeb. Master's thesis, Západočeská univerzita v Plzni, 2020.

[18] *IBM Docs* [online]. IBM, 2019. [cit. 2022/23/01]. Dostupné z: https://www.ibm.com/docs/en/cics-ts/5.1?topic=architecture-web-service-description.

[19] *Web Services Architecture* [online]. Eclipse. [cit. 2021/12/10]. Dostupné z: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis.

[20] *XML introduction* [online]. Mozilla. [cit. 2022/02/20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction.

[21] *YAML Ain't Markup Language (YAML™) version 1.2* [online]. yaml.org. [cit. 2022/02/20]. Dostupné z: https://yaml.org/spec/1.2.2/.

Seznam zkratek

ACM	Assocation for Computing Machinery
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DB	Database
DBLP	Digital Biblipgraphic Library Browser
FTP	File Transport Protocol
FXML	FX Markup Language
GPL	General Public License
HTML	Hypertext Markup Language
HTTP(S)	Hypertext Transfer Protocol (secured)
ID	Identification
IMAP(S)	Internet Message Access Protocol (secured)
JSON	JavaScript Object Notation
KIV	Katedra informatiky a výpočetní techniky
MVC	Model, View, Controller
PDF	Portable Document Format
PDO	PHP Data Objects
PHP	Hypertext Processor
PID	Person Identifier
REST	Representational State Transfer
RPC	Remote Procedure Call
SMTP(s)	Simple Mail Transfer Protocol (secured)
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSDL	Web Service Data Language
WWW	World Wide Web

XML	Extensible Markup Language
XSS	Cross-Site Scripting
YAML	YAML Ain't Markup Language

Přílohy

A Popis adresářové struktury odevzdávaného souboru

- `Text_prace`:
  - **`adresář src`**: obsahuje zdrojové kódy písemné části bakalářské práce,
  - **`A19B0218P_text-prace.pdf`**: PDF soubor písemné bakalářské práce.
- `Aplikace_a_knihovny:`
  - adresář **`app`**: MVC model aplikace,
  - adresář **`css`**: kaskádové styly aplikace,
  - adresář **`img`**: obrázky obsažené v aplikaci,
  - adresář **`doc`**: vygenerovaná PHP dokumentace,
  - adresář **`js`**: JavaScript soubory použité v aplikaci,
  - adresář **`tests`**: PHPUnit třídy,
  - soubor **`bcprace.sql`**: skript pro vytvoření databáze k aplikaci (vytvoření tabulek apod.),
  - soubor **`index.php`**,
  - soubor **`job.php`**: testovací skript integrace do stávajícího systému KIV. Nepouštět na živém provozu! Slouží pro testování na lokálním stroji,
  - soubor **`phpunit.xml`**: konfigurační soubor PHPUnit testů,
  - soubor **`settings.inc.php`**: konfigurační soubor webové aplikace,
  - soubor **`README.md`**: popis jednoho z několika možných postupů nasazení a spuštění aplikace.
- **`Vstupni_data`**: ukázka získaných dat z DBLP API,
- **`Vysledky`**: ukázky výstupních dat webových služeb implementovaných v aplikaci,
- **Readme.txt**: detailní popis aktuální adresářové struktury.