Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Agenda vědeckých výsledků

2016

Martin Kordas

Vedoucí práce: Mgr. Jan Outrata, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Martin Kordas
Název práce:	Agenda vědeckých výsledků
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2016
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Jan Outrata, Ph.D.
Počet stran:	82
Přílohy:	1 CD
Jazyk práce:	český

Bibliograhic info

Author:	Martin Kordas
Title:	Agenda of research results
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2016
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Jan Outrata, Ph.D.
Page count:	82
Supplements:	1 CD
Thesis language:	Czech

Anotace

Webová aplikace umožňuje konfigurovatelnou evidenci záznamů o vědeckých výsledcích (nejen) publikačního typu a jejich komfortní prezentování veřejnosti. Podporuje specializované typy exportů a importů, včetně porovnání dat s vybranými příbuznými databázemi vědeckých výsledků. K provozování aplikace postačuje sada volně dostupných technologií.

Synopsis

Web application provides configurable record keeping and comfortable presentation of research results. It focuses mainly (but not exclusively) on publication types of results. Application supports specialized types of export and import, including comparison of database data with related databases of research results. All technologies needed to run the application are freely available.

Klíčová slova: agenda; věda; výzkum; publikace; informační systém; databáze

Keywords: agenda; science; research; publication; information system; database

Děkuji rodině za podporu a trpělivost v průběhu vytváření bakalářské práce a vedoucímu práce Mgr. Janu Outratovi, Ph.D. za poskytnuté konzultace.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Základní informace

1.1 Evidence vědeckých výsledků

Úloha evidence a prezentace publikační činnosti akademických pracovníků bývá v praxi řešena na několika vzájemně provázaných úrovních. Kromě centralizované celostátní databáze jsou vědecké výsledky evidovány v oddělených informačních systémech jednotlivých vysokých škol a možnost separátní evidence je přitažlivá i pro jejich dílčí organizační celky, jako například katedry.

Vědecké výsledky se v České republice klasifikují na základě druhu (z publikačních druhů výsledků jmenujme např. článek v odborném periodiku, příspěvek ve sborníku konference, kapitolu v knize aj.), přičemž pro každý druh se eviduje odlišný soubor údajů. Některé údaje, jako např. rok uplatnění výsledku nebo jeho původní jazyk, se evidují u všech výsledků současně, zatímco specifické informace, jako např. datum zahájení a zakončení konference, mají význam jen pro výsledky určitého druhu (v tomto případě pro příspěvky ve sbornících konferencí). Mezi evidovanými údaji mají důležitý význam identifikátory umožňující odkázat na záznam o totožném výsledku v některé příbuzné databázi vědeckých výsledků, popř. URL odkazy vedoucí na plný text publikačního výsledku dostupného online.

Název, abstrakt a klíčová slova vědeckého výsledku bývají k dispozici v několika různých jazycích. Každý výsledek může mít přirozeně několik tvůrců, často s různým vztahem k vytvářenému výsledku. Uvedené skutečnosti podstatně ovlivňují strukturování evidovaných dat.

1.2 Požadavky na aplikaci

Cílem této práce je vytvoření webové aplikace evidující vědecké výsledky na úrovni dílčího pracoviště vysoké školy. Zvláštní důraz je kladen na provázanost aplikace s celostátní databází vědeckých výsledků RIV (viz sekci 1.4.1) a s databází OBD UPOL, představující univerzitního protějška celostátní databáze (viz sekci 1.4.2). Předpokládá se evidence především publikačních druhů výsledků (články v odborných časopisech, příspěvky ve sbornících konferencí, atd.).

Podrobné požadavky na funkcionalitu lze rozdělit do několika kategorií, popsaných níže.

Prezentace a interaktivní modifikace dat

- zobrazení dat
- vyhledání dat (v základním a rozšířeném režimu vyhledávání)
- vložení a editace dat (interaktivně formou webového formuláře)
- smazání dat

Hromadná modifikace dat (import)

- import dat z textového souboru s konfigurovatelnou strukturou
- porovnání dat vlastní a příbuzné (externí) databáze s možností importu nalezených rozdílů (tzn. importování nových záznamů pro záznamy přebývající v externí databázi a modifikace existujících pro shodné záznamy), přičemž musí být podporovány tyto externí databáze:
  - RIV (IS VaVaI) (viz sekci 1.4.1)
  - OBD UPOL (viz sekci 1.4.2)

Hromadný export dat

- export dat do textového souboru s konfigurovatelnou strukturou
- export dat v následujících formátech seznamů literatury:
  - BibTEX (viz sekci 1.5.2)
  - vlastní LATEX (viz sekci 1.5.1)
  - DBLP XML (viz sekci 1.5.3)
- export dat ve formátu CSV

Ostatní požadavky

- změnitelnost a rozšiřitelnost sady evidovaných údajů
- veřejné a neveřejné zobrazení dat
- víceuživatelský přístup

1.3 Návrh úrovní oprávnění

Nutnost rozlišovat uživatele na základě úrovně oprávnění plyne přímo z požadavků kladených na aplikaci. Veřejné zobrazení dat nesmí zpřístupnit důvěrné údaje dostupné pouze přihlášeným uživatelům. Samozřejmostí je také přístup pro administrátora s výhradním právem editace a mazání jakýchkoli dat. V zásadě lze tedy rozlišit tři úrovně oprávnění:

1. *nepřihlášený uživatel* – výchozí úroveň na začátku práce s aplikací
2. *přihlášený uživatel* – úroveň nastavená po přihlášení se s uživatelským jménem jiným než „admin“
3. *administrátor* – úroveň nastavená po přihlášení se s uživatelským jménem „admin“

Úrovně jsou vypsány vzestupně podle množství přidělených práv. Až na výjimky platí, že uživatel v určité úrovni oprávnění disponuje přinejmenším stejnými právy jako uživatelé na nižších úrovních.

Práva úrovně *přihlášeného uživatele* je nutné diferencovat podle toho, který uživatel je právě přihlášen. Počítáme-li s víceuživatelským přístupem, musíme modifikaci dat vytvořených jedním uživatelem zakázat pro všechny ostatní uživatele (s výjimkou uživatele s oprávněním *administrátor*). Tím se dostáváme k pojmu *vlastník*. Je-li uživatel *vlastníkem* nějakých dat, pak pouze on spolu s *administrátorem* disponuje právy na jejich modifikaci, resp. smazání. Výchozím *vlastníkem* dat se vždy stává uživatel, který je vytvořit. Dodatečnou změnu *vlastníka* je oprávněn provést pouze *administrátor*.

Mezi jednotlivými úrovněmi oprávnění lze přecházet pouze mezikrokem přes úroveň *nepřihlášeného uživatele* (*přihlášený uživatel* tedy např. nemůže přejít do úrovně *administrátora* pouze v jednom kroku).

Zjednodušený přehled funkcionality, tak jak je dostupná v závislosti na aktuální úrovni oprávnění, poskytuje diagram případů užití (obrázek 1).

1.4 Import dat: příbuzné (externí) databáze

Provázanost s dalšími databázemi pro evidenci publikační činnosti je charakteristickým rysem Agendy vědeckých výsledků. Interakce spočívá ve vyžádání si určitého objemu dat uložených v některé externí databázi a jejich porovnání s daty již existujícími ve vlastní databázi. V návaznosti na to lze provést hromadný import záznamů o výsledcích, jež dosud ve vlastní databázi chyběly anebo údaje o nich nebyly kompletní.

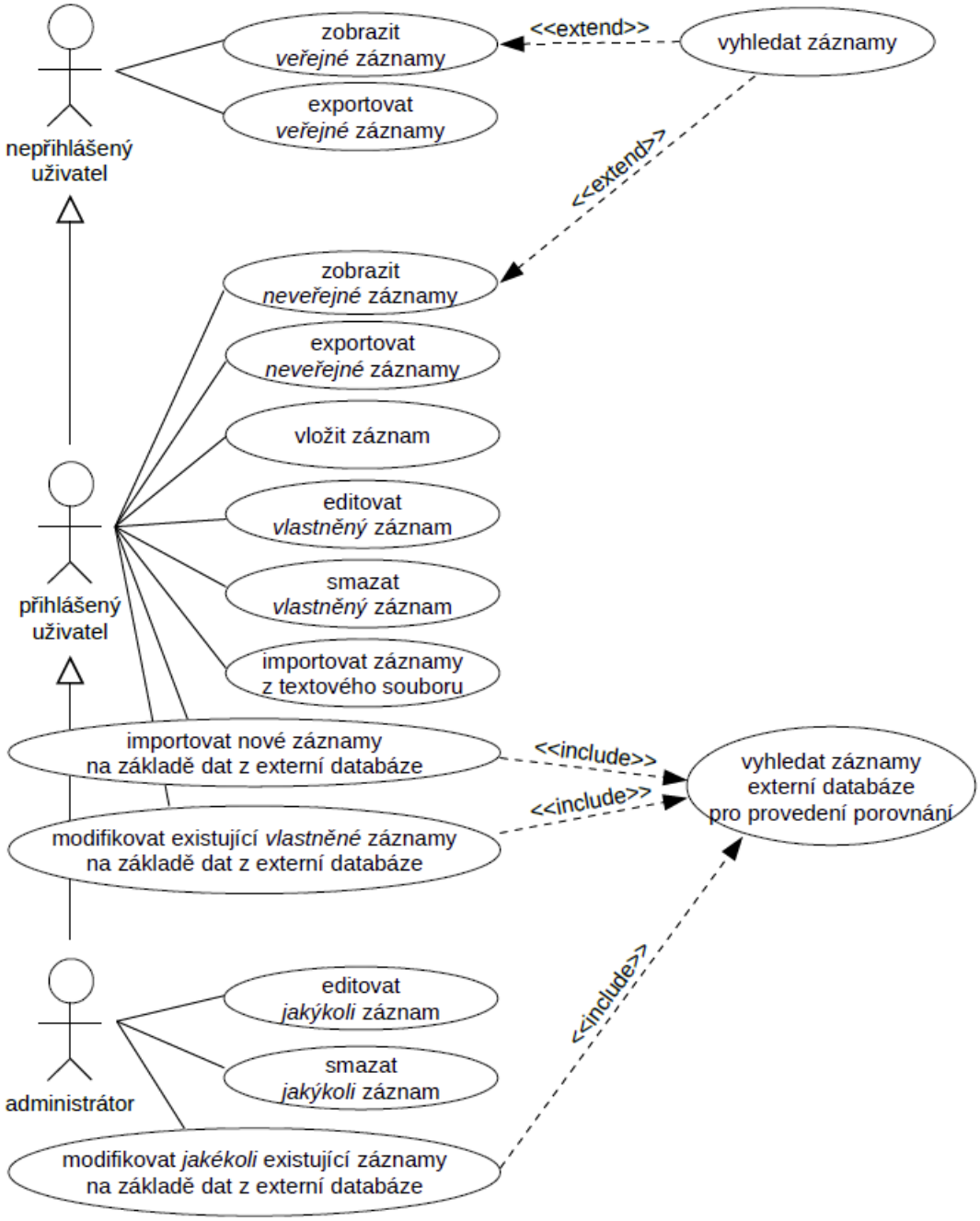
Tuto provázanost má přirozeně smysl implementovat pouze pro ty externí databáze, které evidují nadmnožinu dat evidovaných ve vlastní databázi. Jinak by import dat postrádal smysl. K implementaci byly určeny dvě příbuzné databáze vědeckých výsledků (viz podsekce níže), přičemž další lze doplnit odpovídajícím rozšířením kódu aplikace.

Obě příbuzné databáze vědeckých výsledků, s nimiž je aplikace provázána, jsou v akademickém prostředí často využívány. Pro Agendu vědeckých výsledků tak do určité míry slouží coby předloha uživatelského rozhraní (použití podobných rysů rozhraní usnadňuje uživateli orientaci).

Pro realizaci provázání databází jsou klíčové možnosti exportu externí databáze. Databáze obvykle podporují několik typů exportu, z nichž volíme ten nejsnáze zpracovatelný a informačně nejkomplexnější.

1.4.1 RIV (IS VaVaI)

Rejstřík informací o výsledcích (RIV) představuje celostátní úroveň evidence vědeckých výsledků. RIV je jednou z pěti datových oblastí *Informačního systému výzkumu, experimentálního vývoje a inovací (IS VaVaI)*, jehož úkolem jakožto informačního systému státní správy je „shromažďovat, zpracovávat, poskytovat a využívat údaje o výzkumu, experimentálním vývoji a inovacích podporovaných



Obrázek 1: Diagram případů užití

z veřejných prostředků“ [1]. Samotný RIV pak obsahuje „všechny informace o výsledcích, kterých bylo dosaženo při řešení aktivit výzkumu, experimentálního vývoje a inovací s poskytnutou podporou podle zákona č. 130/2002 Sb. a od roku 2008 i údaje o výsledcích, kterých bylo výzkumnou organizací dosaženo při řešení aktivit výzkumu, experimentálního vývoje a inovací, na které nebyla poskytnuta podpora podle zákona č. 130/2002 Sb“ [1].

Soubor údajů evidovaných v systému RIV se pravidelně aktualizuje. Podrobný přehled o evidovaných údajích a jejich formátu zveřejňuje pro aktuální rok Sekce místopředsedy vlády pro vědu, výzkum a inovace na své webové stránce http://www.vyzkum.cz.

Vyhledávat ve veřejně přístupných údajích systému RIV může kdokoliv prostřednictvím informačního systému na adrese http://www.isvav.cz.

Možnosti exportu Podporován je export do formátů **DBF** a **XLS**. Pro strojové zpracování v Agendě vědeckých výsledků byl zvolen formát **DBF**, neboť obsahuje údaje v holé formě (např. třípísmenné kódy jazyků namísto uživatelských popisků) a součástí exportu je i soubor s metadaty. Pro získání exportů nelze využít žádné strojové rozhraní. Jedinou cestou je tedy strojové zaslání stejných HTTP požadavků, jaké zasílá prohlížeč při práci s klasickým webovým rozhraním.

1.4.2 OBD UPOL

„K evidenci publikační činnosti je na Univerzitě Palackého v Olomouci využíván program s označením *Osobní bibliografická databáze* (OBD)“. [2] „V OBD musí být evidována veškerá publikační činnost zaměstnanců a studentů UP pro potřeby výkaznictví – především pro evidenci Rejstříku informací o výsledcích.“ [2]

Webová aplikace umožňující veřejný a neveřejný přístup k údajům o výsledcích se nachází na adrese http://obd.upol.cz.

Možnosti exportu Funkce exportu je dostupná pouze v neveřejném rozhraní webové aplikace. Podporovány jsou různé varianty exportů do **HTML**, export do **XML** a export do **CSV**. Pro strojové zpracování se nejvíce hodí export do **XML**, protože obsahuje nejvíce údajů a je strukturovaný. Pro strojové získání exportů je nutné simulovat obvyklé HTTP požadavky prohlížeče (strojové rozhraní není k dispozici).

1.5 Export dat: bibliografické formáty

Většinu dat aplikace by v praxi měly tvořit záznamy o vědeckých výsledcích publikačního rázu. Pro budoucí správné pochopení funkce exportu záznamů do specializovaných formátů seznamů literatury nyní tyto formáty popíšeme.

1.5.1 Vlastní LaTeX

Jde o seznam literatury typografického systému LaTeX vytvářený pro aktuální dokument ručně v prostředí `thebibliography`. Jednotlivé bibliografické záznamy začínají makrem `bibitem{klíč}`. Jinak formát není omezen. V textu lze používat klasická makra LaTeXu. Více viz [3] a příklad ve zdroj. kódu 1.

```
1 \begin{thebibliography}{9}
2   \bibitem{knu86} Donald~E.~Knuth. \newblock {\em The TeXbook}.
        \newblock Addison-Wesley Professional, 1986.
3   \bibitem{lam94} Leslie~A.~Lamport. {\em LaTeX: A~Document
        Preparation System}, 2nd edition. Addison-Wesley Professional,
        1994.
4 \end{thebibliography}
```

Zdrojový kód 1: Ukázka seznamu literatury pro vlastní LaTeX

Z důvodu velké variability tohoto formátu je na místě přímá konfigurace tvaru exportu uživatele.

1.5.2 BibTeX

BibTeX je název programu umožňujícího pokročilou práci s bibliografickými údaji v systému LaTeX. Veškeré údaje o publikacích nese soubor typu **BIB**, což je textový soubor s pevně danou syntaxí.

Každé publikaci zaznamenané v souboru **BIB** je přiřazen určitý *BibTeX typ* (v syntaxi **BIB** souboru je uvozen zavináčem), např. `article` pro článek v časopisu nebo `book` pro knihu. Každému *BibTeX typu* přísluší sada *BibTeX polí*, čili údajů, které se pro daný typ publikace evidují. *BibTeX pole* se pak podle dobrovolnosti uvedení dělí na povinná a volitelná. Podrobný přehled všech *BibTeX typů* a jim odpovídajících *BibTeX polí* obsahuje [4], eventuálně [5] (pro rozšířený formát BibLaTeX). Příklad obsahu **BIB** souboru obsahuje zdroj. kód 2.

Zpracování údajů v **BIB** souboru závisí na použitém BibTeX citačním stylu. Specifické citační styly mohou *BibTeX typy* asociovat s *BibTeX poli* pozměněným způsobem. Pro specifický citační styl, jakým může být např. styl pro normu *ČSN ISO 690*, je tedy požadován specifický tvar exportu.

1.5.3 DBLP XML

Formát je spjat s bibliografickou databází *DBLP computer science bibliography*, jež má svůj původ na univerzitě v německém Trevíru. K databázi lze přistupovat přes veřejné webové rozhraní na adrese http://dblp.uni-trier.de.

DBLP XML je klasický **XML** soubor se strukturou připomínající seznamy literatury *BibTeXu*. Jednotlivým bibliografickým záznamům je přiřazen typ a na jeho základě se doplní další vyžadované informace. Typ bibliografického záznamu se v syntaxi *DBLP XML* uvádí v názvu elementu nesoucího daný záznam

```
@book{knu86,
  author = {Donald E. Knuth},
  title = {The TeXbook},
  publisher = {Addison-Wesley Professional},
  year = {1986}
}
@book{lam94,
  author = {Leslie A. Lamport},
  title = {LaTeX: A~Document Preparation System},
  publisher = {Addison-Wesley Professional},
  year = {1994}
}
```

Zdrojový kód 2: Ukázka seznamu literatury ve formátu BibTEXu

(např. elementy `<article>` pro článek v časopisu a `<book>` pro knihu) a podrobné údaje obsahují potomoci tohoto elementu (např. elementy `<author>` nebo `<title>`). Pro publikaci s více autory lze použít několik samostatných elementů `<author>`. Příklad *DBLP XML* souboru obsahuje zdroj. kód 3.

```
<dblp>
  <book key="knu86">
    <author>Donald E. Knuth</author>
    <title>The TeXbook</title>
    <publisher>Addison-Wesley Professional</publisher>
    <year>1986</year>
  </book>
  <book key="lam94">
    <author>Leslie A. Lamport</author>
    <title>LaTeX: A~Document Preparation System</title>
    <publisher>Addison-Wesley Professional</publisher>
    <year>1994</year>
  </book>
</dblp>
```

Zdrojový kód 3: Ukázka seznamu literatury ve formátu *DBLP XML*

Validitu bibliografického **XML** lze ověřit s pomocí souboru **DTD**[1]. Pro další podrobnosti o formátu viz [6].

[1] **DTD** pro *DBLP XML* je k dispozici na http://dblp.uni-trier.de/xml/dblp.dtd.

2 Technická dokumentace

2.1 Použité technologie

Koncepce Agendy vědeckých výsledků coby webové aplikace pochází ze samotného zadání aplikace. Webové rozhraní umožňuje snadný přístup nejen interním uživatelům, ale i široké veřejnosti. Pro vývoj byl použit soubor široce známých a bezplatně dostupných technologií.

MariaDB

Relační databázový systém *MariaDB* vzešel ze systému *MySQL* a zachovává s ním vzájemnou kompatibilitu. Na serveru zajišťuje veškeré čtení a modifikaci dat. Pro dotazy na databázi se využívá standardizovaný jazyk *SQL* s některými rysy specifickými pouze pro *MariaDB*, resp. *MySQL*.

PHP

PHP je víceúčelový skriptovací jazyk vhodný zvláště pro vývoj webových aplikací [7]. Syntakticky vychází z jazyka *C*. *PHP* kód je interpretován na serveru, kde zajišťuje dynamické generování HTML stránek a obsluhu požadavků na modifikaci databázových dat. Tvoří tak jádro celé aplikace. Od verze 5 disponuje jazyk pokročilým objektovým systémem; při vývoji aplikace byl objektově orientovaný přístup ve velké míře využit.

Součástí kódu aplikace je nezkompilovaná externí knihovna **PHP XBase**[2], využívaná pro čtení **DBF** souborů (standardní *PHP* rozšíření **dBase** nelze pro daný typ **DBF** souborů[3] použít). Knihovna **PHP XBase** se nachází ve složce `src/class/XBase`.

JavaScript

JavaScript zajišťuje nutné skriptování na straně klienta, tj. v klientově prohlížeči. Snazšímu ovládání aplikace a větší efektivitě napomáhá užití technologie asynchronních požadavků **AJAX**.

Javascriptová knihovna **jQuery**[4] je zahrnuta především pro správnou funkčnost frameworku *Bootstrap* (viz níže). Ve vlastních kódech Agendy vědeckých výsledků se její funkcionalita používá spíše výjimečně.

Plugin **Twitter Bootstrap Hover Dropdown**[5] rozšiřuje možnosti rozbalovacích nabídek frameworku *Bootstrap* (viz níže).

Všechny externí javascriptové knihovny se nachází ve složce `src/script`.

[2] Podrobnosti na http://github.com/hisamu/php-xbase.

[3] Jedná se o **DBF** soubory typu *Visual FoxPro*. V tomto formátu exportuje data systém RIV (IS VaVaI).

[4] Podrobnosti na http://www.jquery.com.

[5] Podrobnosti na http://github.com/CWSpear/bootstrap-hover-dropdown.

Bootstrap

Jedná se o CSS a javascriptový framework usnadňující tvorbu graficky působivých webových stránkek.[6] Zvláštní důraz je kladen na responzivní design[7]. Součástí frameworku je také sada ikon Glyphicons Halflings[8]. Soubory frameworku se nachází ve složce `src/bootstrap`.

Pohnutkou pro použití frameworku byl soulad designu aplikace s designem webu katedry, která bude aplikaci využívat. Responzivita designu aplikace, ve frameworku implicitně přítomná, je víceméně vedlejším produktem, protože přístup k aplikaci z přenosných zařízení se předpokládá ojediněle.

2.2 Návrh databáze

2.2.1 Struktura dat

Základní entitou evidovanou v aplikaci je záznam o vědeckém výsledku (dále jen ***záznam***).

Každý ze záznamů disponuje názvem, abstraktem a klíčovými slovy, to vše obvykle v několika jazycích. Proto zavádíme entitu s názvem ***titul***, jež nese údaje o názvu, abstraktu a klíčových slovech záznamu pro jeden konkrétní jazyk. Vztah mezi záznamy a tituly je typu **1:M**.

Třetí entitou je ***autor***. Platí, že jednomu záznamu může být přiřazeno více autorů a zároveň jeden autor může být použit pro více záznamů. Vztah mezi záznamy a autory je tedy typu **M:N**. Zároveň platí, že některé informace týkající se autora nelze evidovat pro autora obecně (tak jako jméno a příjmení), nýbrž vždy jen ve vztahu k záznamu, jemuž byl autor přiřazen (to se týká např. pořadí autora mezi ostatními autory záznamu nebo jeho mentálního podílu na vzniku záznamu).

Uvedený koncept tří databázových entit dobře odpovídá struktuře **XML** exportu databáze OBD UPOL, s níž má být naše aplikace provázána.

Grafickým znázorněním konceptu získáme Entity-Relationship (E-R) diagram (obr. 2).

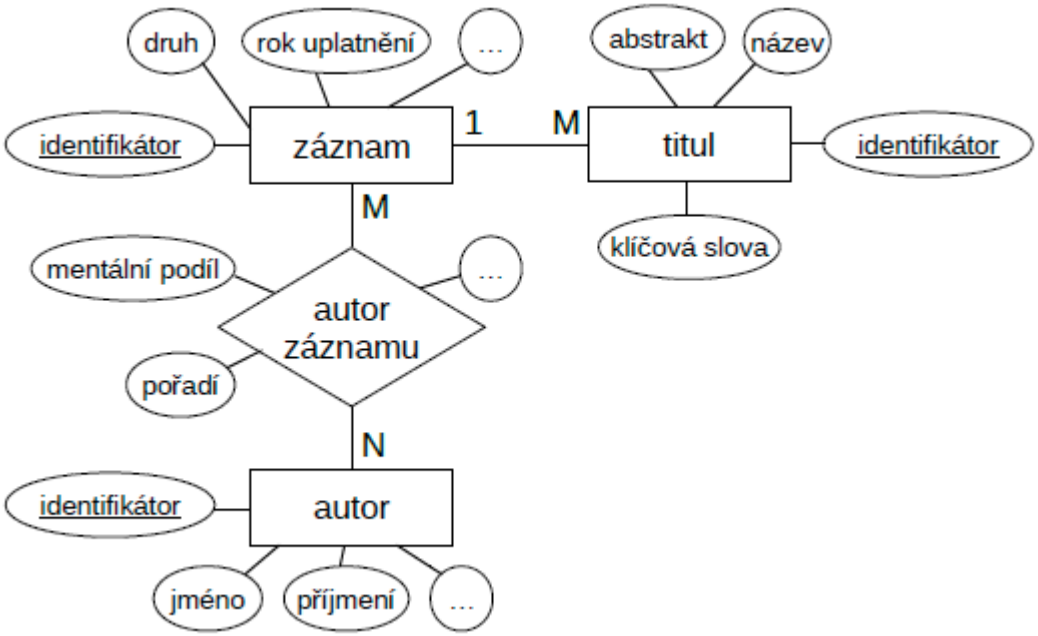
Zavedená terminologie pro označování databázových entit má své rezervy. Jak už bylo popsáno výše, *titul* není označení výhradně pro název vědeckého výsledku, nýbrž pro soubor údajů o názvu, jazyku a klíčových slovech vědeckého výsledku. Entita autor zase nepředstavuje výhradně autora vědeckého výsledku, nýbrž jeho tvůrce s obecně různým (nastavitelným) vztahem k vytvářenému výsledku.

Mluvíme-li o souboru nějakých dat bez ohledu na to, co obsahují, používáme pojem *položka* (např. počet položek zobrazených v tabulce).

[6] Podrobnosti na http://www.getbootstrap.com.

[7] Responzivní design umožňuje přizpůsobit rozvržení stránky proporcím zařízení, na kterém je právě zobrazována.

[8] Podrobnosti na http://www.glyphicons.com.



Obrázek 2: Entity-Relationship (E-R) diagram

2.2.2 Databázové schema

Jak ukazuje tab. 1, můžeme na základě E-R diagramu odvodit čtyři databázové tabulky. Tři z databázových tabulek odpovídají E-R entitám a vazební tabulka autorů odpovídá E-R vztahu *autor záznamu*. Pátou tabulkou (do E-R diagramu nezahrnutou) je tabulka uživatelů, sloužící pro potřeby víceuživatelského přístupu a autentizace. Uvnitř aplikace je na tabulky odkazováno pomocí krátkého kódu (tento kód budeme využívat i pro účely této dokumentace), případně přímo databázovým názvem uloženým v některé PHP konstantě `TABLE_*`.

E-R entita (vztah)	**Jméno tabulky**	**Kód tabulky**	**PHP konstanta**
Záznam	Tbl. záznamů	*„data“*	`TABLE_DATA`
Titul	Tbl. titulů	*„title“*	`TABLE_TITLE`
Autor	Tbl. autorů	*„author“*	`TABLE_AUTHOR`
Autor záznamu	Vazební tbl. autorů	*„data_author“*	`TABLE_DATA_AUTHOR`
—	Tabulka uživatelů	*„users“*	`TABLE_USERS`

Tabulka 1: Přehled databázových tabulek

Interní identifikátory atributů Není žádoucí, aby v kódu aplikace figurovaly pevně určené databázové názvy atributů tabulek. To by vyžadovalo rozsáhlé změny kódu při každé změně databázového názvu některého atributu. Proto

v aplikaci odkazujeme na atributy pomocí zobecnělých *interních identifikátorů*, jejichž mapování na skutečné databázové názvy atributů definují a také provádějí PHP třídy `Columns` a `UsersColumns`. Názvy interních identifikátorů začínají obvykle předponou *i_*.

Interní identifikátor atributu ve spojení s kódem tabulky přehledně identifikují konkrétní databázový atribut. V této dokumentaci, stejně jako v komentářích ve zdrojových kódech, používáme identifikaci atributů ve tvaru *i_interni_identifikator („kod_tabulky“)*, čili např. *i_id („data“)* pro atribut identifikátor tabulky záznamů.

Pevné a variabilní atributy Ty atributy databázového schematu, které mají zásadní význam pro funkcionalitu a jejichž smazání anebo typová úprava nepřichází v úvahu, označujeme jako ***pevné atributy***. Jejich databázové názvy obvykle začínají předponou *fix_*.

V aplikaci zavádíme tyto pevné atributy (popsány odpovídajícím interním identifikátorem):

- *i_id* – identifikátor položky
- *i_username* – vlastník položky
- *i_public* – veřejnost položky
- atributy tabulky uživatelů (viz níže)

Ve dvou různých tabulkách mohou být pevné atributy se stejným interním identifikátorem. V takovém případě jde o zcela nezávislé databázové atributy, ale v rámci svých tabulek plní obdobnou funkci.

Atributy mající pro funkčnost aplikace naopak menší anebo zcela postradatelný význam označujeme jako ***variabilní atributy***. Tyto atributy lze definovat v libovolném počtu a s volitelnými parametry prostřednictvím konfiguračního souboru příslušné tabulky (viz sekci 4.3.2). Konf. soubor poskytuje aplikaci veškeré informace o tom, jak s variabilními atributy zacházet. Databázové názvy variabilních atributů obvykle začínají předponou *var_*.

Variabilním atributům obvykle nepřísluší interní identifikátor, protože aplikace se na ně odkazuje na základě informací v konfiguračním souboru. Výjimku tvoří variabilní atributy se zvláštním významem. Jako příklad může posloužit atribut *i_riv („data“)*, nesoucí kód identifikující záznam o totožném vědeckém výsledku v databázi RIV (IS VaVaI). Při výpisu aplikace nezobrazí prostou hodnotu atributu jako v případě běžných atributů, nýbrž odkaz do webového rozhraní systému RIV (IS VaVaI). Díky internímu identifikátoru *i_riv* aplikace pozná, který z variabilních atributů definovaných v konfiguračním souboru vyžaduje tento způsob výpisu.

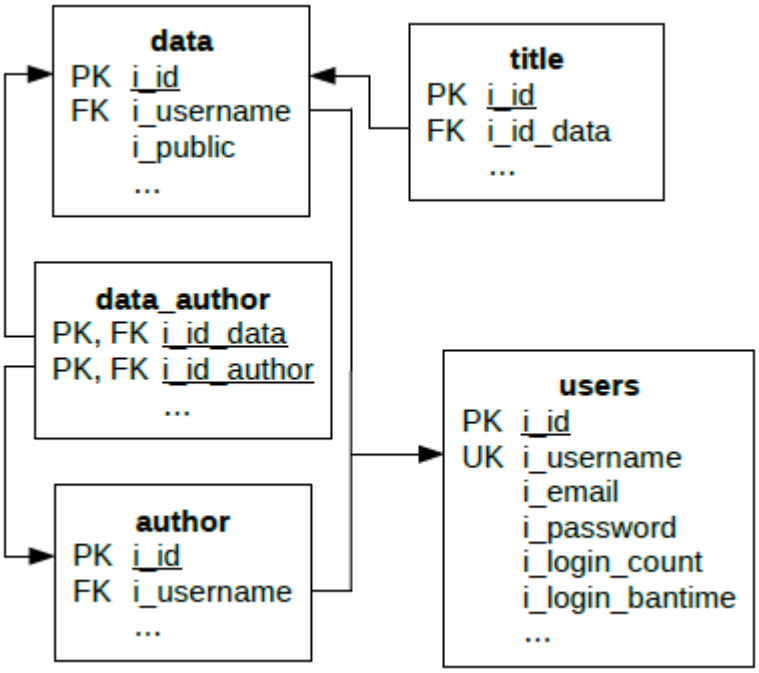
Pro přehled interních identifikátorů odkazujících na variabilní atributy viz PHP třídu `Columns` a všechny tam uvedené interní identifikátory vyjma těch odkazujících na pevné atributy.

S varabilními atributy, na něž odkazuje nějaký interní identifikátor, nesmí být manipulováno tak volně jako s ostatními variabilními atributy. (Jejich smazání by způsobilo selhání aplikace a doporučena nabývá ani změna datového typu.) Pro podrobnosti viz sekci 4.3.2 (část o atributech se zvláštním významem).

Atributy tabulky uživatelů Databázové schema tabulky uživatelů není konfigurovatelné, a proto obsahuje pouze pevné atributy. Jsou jimi:

- *i_id* – identifikátor uživatele
- *i_username* – jméno uživatele
- *i_password* – heslo v hashovaném tvaru
- *i_email* – kontaktní email
- *i_login_count*, *i_login_bantime* – údaje pro funkci dočasného zablokování přihlašování (po překročení maximálního počtu neúspěšných přihlášení)

Na základě uvedených informací jsme již schopni sestavit finální model databáze (obr. 3). Pro jednoduchost model znázorňuje pouze pevné atributy. Namísto názvů tabulek jsou uvedeny jejich kódy (viz tab. 1).



Obrázek 3: Model databáze

Z modelu plyne, že veřejnost (atribut *i_public*) je evidována pouze pro entitu záznamu (tabulka *„data“*). Veřejnost autorů a titulů jednotlivých záznamů je dána veřejností záznamů, k nimž náleží.

Každý záznam v tabulce *„data“* má nějakého vlastníka (atribut *i_username*)[9]. Datům entity titul (tabulka *„title“*) vlastníka nepřiřazujeme, protože jejich nezměnitelným vlastníkem je vždy vlastník záznamu, ke kterému entita titulu náleží. Totéž platí o datech tabulky *„data_author“*, reprezentující údaje o autorovi ve vztahu ke konkrétnímu záznamu. Naopak tabulka *„author“* již atribut vlastníka obsahuje, protože vlastníkem údajů týkajících se autora obecně je specifický uživatel, který tohoto autora vytvořil.

Přesné SQL příkazy s definicemi databázových tabulek a jejich pevných atributů obsahuje soubor `src/sql/create_table.sql`. Variabilní atributy lze ve schematu vytvářet automatizovaně na základě informací v konfiguračním souboru příslušné tabulky (nástroj administrátora aplikace – viz sekci 4.6.1).

2.2.3 Přístupová oprávnění

Z bezpečnostních důvodů používá aplikace pro přístup k databázi systém oprávnění, který je analogický k úrovním oprávnění používaným v samotné aplikaci (viz sekci 1.3). Každé ze tří úrovní oprávnění přísluší odlišný databázový uživatel s odpovídajícími právy. Čtvrtý db. uživatel pak slouží pro potřeby autentizace, tj. ve chvíli, kdy totožnost uživatele (a tím ani úroveň oprávnění) není ještě známa. Více viz direktivy `username_*`, `password_*` v konf. souboru `src/conf/application.ini.php`. Pro SQL příkazy vytvoření a přidení práv db. uživatelům viz soubor `src/sql/create_grant_user.sql`).

V případě nedostupnosti většího množství db. uživatelů lze použít jediného uživatele s přidělenými oprávněními pokrývajícími všechny čtyři úrovně oprávnění.

2.2.4 Další specifika

Prázdné hodnoty řetězcových datových typů jsou vždy reprezentovány jako prázdné hodnoty `NULL`, nikoli jako prázdné řetězce.

Výčtový typ `ENUM` neobsahuje nikdy prázdnou hodnotu `NULL`. Validní výčtovou hodnotou však může být prázdný řetězec (tato výčtová hodnota má v aplikaci zvláštní význam neurčené hodnoty).

2.3 Adresářová struktura aplikace

Zdrojové kódy a další soubory nutné pro fungování aplikace obsahuje složka `src` na přiloženém CD. Přímo ve složce `src` jsou umístěny PHP skripty zajišťující dynamické generování HTML stránek a obecně obsluhující požadavky uživatele. Další soubory jsou obsaženy v těchto vnořených složkách:

- `bootstrap` – CSS a javascriptový framework (více v sekci 2.1).
- `class` – PHP třídy, implementující převážnou část funkcionality PHP skriptů.

[9] Pro význam pojmu *vlastník* viz sekci 1.3.

  - `class/data` – čtení a zápis dat vlastní databáze a importních dat (více v sekci 2.5.3).
    * `class/data/dbhandler` – třídy pro nízkoúrovňové čtení dat získaných z databáze.
  - `class/lib` – třídy používané napříč celou aplikací.
    * `class/lib/auth` – autentizační třídy (více v sekci 2.5.2).
    * `class/lib/conn` – databázové třídy (více v sekci 2.5.2).
  - `class/XBase` – externí knihovna pro čtení **DBF** souborů (více v sekci 2.1).
- `conf` – konfigurační soubory.
  - `conf/biblio` – konfigurační soubory bibliografických exportů (více v sekci 4.3.6).
  - `conf/textfile_templates` – šablony pro import a export do textového souboru, vč. šablony bibliografického exportu pro vlastní LaTeX (více v sekci 4.3.7).
- `img` – obrázky.
- `load` – PHP skripty zajišťující načtení globálního prostředí programu (více v sekci 2.5.1).
- `script` – vlastní a externí javascriptové knihovny.
- `style` – kaskádové styly.
- (`sql` – SQL příkazy pro počáteční vytvoření databáze a databázových uživatelů. Tato složka se nekopíruje na webový server.)

Konvence pojmenování PHP skriptů

- Pokud PHP skript negeneruje HTML výstup (pouze provádí nějaký příkaz klienta), jeho název nese jedno z těchto zakončení:
  - `*_script.php`
  - `*_hscript.php`[10] (manipulace s vybranou položkou, příp. obsluha různých požadavků na konkrétní stránce)
  - `*_multi_hscript.php` (hromadná manipulace s položkami)
- Slouží-li PHP skript jako odpověď pro AJAX požadavek, jeho název nese zakončení `*_ajax.php`.

[10] „h“ jako „handle“

2.4 Důležité konstanty

Kromě úpravy konfiguračních souborů lze fungování aplikace ovlivnit i uvážlivou změnou konstant ve zdrojových kódech.

2.4.1 Globální PHP konstanty

Jsou definovány v souboru `src/load/constants.php`. Výčet nejdůležitějších:

- `PROVIDER_AUTH` (string) – název používané autentizační třídy. Více v sekci 2.5.2.
- `PROVIDER_CONN` (string) – název používané databázové třídy. Více v sekci 2.5.2.
- `LINE_BREAK` (string) – posloupnost znaků reprezentujících odřádkování v hodnotách ukládaných do databáze a v exportovaných souborech. Import a export do textového souboru používá oddělenou sadu konstant ve třídě `Text`[11].
- `LINE_BREAK_PCRE` (string) – jako `LINE_BREAK`, ale řetězec je určen pro použití v regulárních výrazech funkcí knihovny PCRE.
- `IMCHAR_IMPORT_LINE_WIDTH` (integer) – maximální šířka řádku pro importní hodnoty víceřádkového řetězcového typu *imchar*. V případě překročení limitu budou při importu doplněna odřádkování.
- `ERROR_REPORTING` (integer) – bitová maska určující úroveň zobrazovaných chyb (viz PHP direktivu *error_reporting*). (Chyby se zobrazují pouze při zapnuté direktivě *debug_mode* konf. souboru `src/conf/application.ini.php`.)

2.4.2 Konstanty PHP tříd

Konstanty s omezeným rozsahem využití nalezneme v PHP třídách. Mezi nejdůležitější patří:

- `HTTPDigestAuth::SESSION_TIME_LIMIT` (integer) – maximální doba nečinnosti do automatického odhlášení. Uvádí se v sekundách. (Pouze při použití autentizační třídy `HTTPDigestAuth`.)
- `HTTPDigestAuth::LOGIN_COUNT_LIMIT` (integer) – maximální počet neúspěšných pokusů o přihlášení pod určitým uživatelským jménem. Po překročení limitu dojde k dočasném zablokování přihlašování. (Pouze při použití autentizační třídy `HTTPDigestAuth`.)

[11] Při exportu do textového souboru proto odřádkování určená definicí šablony reprezentuje řetězec `Text::LINE_BREAK` a odřádkování uvnitř exportních hodnot řetězec `LINE_BREAK`. Řetězce v konstantách by měly být stanoveny konzistentně.

- `HTTPDigestAuth::LOGIN_BANTIME_LIMIT` (integer) – doba dočasného blokování přihlašování. Uvádí se v sekundách. (Pouze při použití autentizační třídy `HTTPDigestAuth`.)
- `HTTPDigestAuth::LOGIN_REDIRECT` (string, bool) – název skriptu, kam bude uživatel přesměrován po úspěšném přihlášení, nebo logická nepravda pro variantu bez přesměrování. (Pouze při použití autentizační třídy `HTTPDigestAuth`.)
- `HTTPDigestAuth::FORCE_HTTPS` (bool) – určuje, má-li být pro každý přístup k aplikaci vynucen šifrovaný přenos protokolem HTTPS. (Pouze při použití autentizační třídy `HTTPDigestAuth`.)
- `MySQLiConn::TRANSACTION_ISOLATION_LEVEL` (string) – úroveň izolace transakčního zpracování. (Pouze při použití databázové třídy `MySQLiConn`.)
- `MySQLiConn::PERSISTENT_CONNECTION` (bool) – určuje, má-li být připojení k databázi persistentní. (Pouze při použití databázové třídy `MySQLiConn`.)
- `MySQLiConn::SERVER` (string) – jméno nebo IP adresa databázového serveru. (Pouze při použití databázové třídy `MySQLiConn`.)
- `ODBCConn::TRANSACTION_ISOLATION_LEVEL` (string) – úroveň izolace transakčního zpracování. (Pouze při použití databázové třídy `ODBCConn`.)
- `ODBCConn::DSN` (string) – jméno *Data Source Name (DSN)* popisujícího připojení k databázi přes rozhraní ODBC. Konfigurace DSN se na různých systémech provádí odlišně; jako znakovou sadu se doporučuje použít UTF-8. (Pouze při použití databázové třídy `ODBCConn`.)
- `ExportBiblioBibtex::FILENAME_SEND` (string),
  `ExportBiblioDBLP::FILENAME_SEND` (string),
  `ExportCSV::FILENAME_SEND` (string),
  `ExportText::FILENAME_SEND_DEFAULT` (string) – výchozí název exportovaného souboru.
- Konstanty třídy `Text` určující způsob odřádkování při importu a exportu do textového souboru. Pro více informací viz komentář ve zdrojovém kódu.
- `FetchText::PATTERN_REPETITION_REDO` (bool) – ovlivňuje způsob čtení importního textového souboru. Více v sekci 4.3.7.3.

2.4.3 Globální proměnné JavaScriptu

Významné jsou pouze globální proměnné v souboru `src/script/common_script.js`, určující parametry vyskakovacích oken (řetězec s parametry se předává javascriptové funkci `Window.open()`):

- `WSPECS_TITLE` (string) – okno detailu, vložení a editace titulu záznamu.
- `WSPECS_DATA_AUTHOR` (string) – okno detailu, vložení a editace autora záznamu.
- `WSPECS_LATEX_TEMPLATE` (string) – okno pro úpravu šablony bibliografického exportu pro vlastní LaTeX.
- `WSPECS_DEFAULT` (string) – ostatní vyskakovací okna.

2.5 Vlastní PHP aplikace

Popis aplikace se bude ve většině případů týkat PHP tříd (umístěny ve složce `src/class`), které zapouzdřují většinu funkcionality.

2.5.1 Globální prostředí programu

Na začátku každého skriptu dochází prostřednictvím souboru `src/load/load.php` k zavedení programového prostředí. Toto prostředí sestává z globálních proměnných a konstant definovaných v souboru `src/load/constants.php` a globálních funkcí definovaných v souboru `src/load/functions.php`. Dále je ovlivněno vedlejším efektem funkcí volaných v souboru `src/load/load.php`.

Konstanty definované v souboru `src/load/constants.php` popisuje sekce 2.4. Z funkcí definovaných v souboru `src/load/functions.php` stojí za zmínku:

- `agenda_to_db()` – převod hodnoty z formátu používaného v aplikaci na formát používaný v databázi. Funkce by měla být volána pro všechny hodnoty použité v dotazu na databázi[12].
- `db_to_agenda()` – převod hodnoty z formátu používaného v databázi na formát používaný v aplikaci. Funkce by měla být volána pro všechny hodnoty získané z databáze.
- `log_error_*()` – sada funkcí zapisujících do logovacího souboru chyby knihoven, které nepodporují implicitní hlášení o chybách.

2.5.2 Systémová třída (Connection)

Na začátku každého skriptu bývá vytvořena instance tzv. *systémové třídy* `Connection`. Třída umožňuje provedení autentizace, získání informace o přihlášenosti uživatele a volitelně i přístup k databázi. Tuto funkcionalitu třída deleguje tzv. *autentizační třídě* (instance umístěna v proměnné `Connection::$auth`) nebo případně *databázové třídě* (instance umístěna v proměnné `Connection::$conn`).

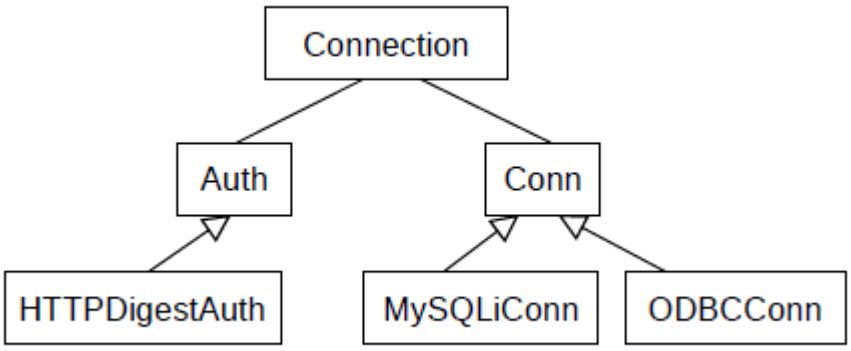
[12] Z historických a praktických důvodů dochází k vynechání volání funkce. V případě vynechání volání musí platit, že pro interní datový typ převáděné hodnoty není žádný převod definován.

Konkrétní třídy, kterým bude delegováno provedení autentizace a připojení k databázi, jsou určeny konstantami (`PROVIDER_AUTH` se jménem autentizační třídy a `PROVIDER_CONN` se jménem databázové třídy). Aplikaci lze tedy provozovat s různými způsoby provedení autentizace a s využitím různých knihoven pro připojení k databázi. Všechny autentizační třídy rozšiřují abstraktní třídu `Auth` a jsou uloženy ve složce `src/class/lib/auth`. Databázové třídy rozšiřují abstraktní třídu `Conn` a nachází se ve složce `src/class/lib/conn`.

Změna používané autentizační nebo databázové třídy může kromě přepsání příslušné konstanty vyžadovat i provedení dalších kroků, popsaných ve zdrojových kódech odpovídající abstraktní třídy `Auth` nebo `Conn`. Metoda autentizace může klást omezení i na používané autentizační údaje (rozsah podporovaných znaků).

Ať už je používána jakákoli autentizační třída, pro jméno uživatele nelze z technických důvodů použít prázdný řetězec a řetězec „0“.

Ve výchozí implementaci je k dispozici jedna autentizační třída a dvě třídy databázové, jak ukazuje diagram tříd na obr. 4. Výchozí autentizační třídou je tedy `HTTPDigestAuth`, výchozí databázovou třídou `MySQLiConn`.



Obrázek 4: Systémová třída Connection a související třídy

Výběr autentizační třídy Pro výběr autentizační třídy je podstatné, jakým způsobem chceme autentizační údaje zasílat a ověřovat. Výchozí autentizační třída `HTTPDigestAuth` předpokládá šifrované zaslání autentizačních údajů v HTTP hlavičce a jejich ověření vůči údajům uloženým přímo v databázi aplikace (tabulka uživatelů). Zobrazení formuláře pro zadání autentizačních údajů obstarává prohlížeč. Pro informace o dalších rysech autentizační třídy (automatické odhlašování, ochrana před brute-force útokem, vynucení HTTPS aj.) viz komentáře ve zdrojovém kódu. Konfigurace činnosti třídy se provádí prostřednictvím konstant třídy, popsaných v sekci 2.4.2.

Nově implementovaná alternativní autentizační třída by v kontrastu k výchozí `HTTPDigestAuth` mohla podporovat zasílání autentizačních údajů klasickou metodou POST[13], příp. jejich ověření s pomocí externí autentizační služby.

Výběr databázové třídy Databázová třída určuje knihovnu, s jejíž pomocí budou dotazována databázová data. Výchozí třída `MySQLiConn` využívá knihovnu **MySQLi**, určenou výhradně pro databázový server MySQL, zatímco alternativní třída `OBDCConn` pracuje s API knihovny **ODBC**, použitelným pro komunikaci s různými databázovými servery. Další informace o fungování tříd obsahují komentáře ve zdrojových kódech. Konfigurace činnosti tříd se provádí prostřednictvím třídních konstant, popsaných v sekci 2.4.2.

Na první pohled praktická možnost záměny stávající databázové třídy za třídu používající jinou databázovou knihovnu s sebou nese nutnost omezení společného rozhraní databázových tříd na nejnutnější operace podporované všemi databázovými knihovnami.

Přihlašovací údaje, které databázová třída použije pro připojení k databázi, závisí na úrovni oprávnění aktuálního uživatele (Viz funkci `Conn::resolve_credentials()`). Připojení k databázi pro *přihlášeného uživatele* proto např. neumožňuje provádět činnosti určené výhradně *administrátorovi*.

2.5.3 Práce s daty

Pro práci s databázovými daty nebo daty rozpracovaného záznamu, uloženými v session, jsou na nižší úrovni aplikace určeny třídy ve složce `src/class/data`.

Některé ze tříd implementují následující rozhraní:

- `IDataGet` – jednoduché čtení sady dat. Rozhraní rozšiřuje nativní rozhraní `Countable`, umožňující použít objekt jako argument nativní funkce `count()`, a nativní rozhraní `Iterator`, díky kterému lze přes objekt iterovat cyklem `foreach`.
- `IDataSet` – jednoduchý zápis (příp. úprava) datové položky.
- `IDataGetMulti` – komplexní čtení dat (k dispozici jsou všechna data spjatá s konkrétní záznamem, tj. i všichni jeho autoři a tituly). Narozdíl od `IDataGet` neiterujeme přes objekty implementujících tříd cyklem `foreach` (třídy neobsahují vnitřní pointer, který by se dal posouvat), ale informace o pozici žádaných dat předáváme v argumentech funkce.

Přehled tříd:

- `DataGetDatabase` (implementuje `IDataGet`) – jednoduché čtení sady dat získaných z databáze. Většina funkcionality je delegována objektu uloženému v proměnné `IDataGet::$handler`, což je objekt třídy implementující

[13] To by vyžadovalo vytvoření vlastního HTML formuláře pro zadání údajů.

rozhraní `IDbHandler` (rozhraní i implementující třídy se nachází ve složce `src/class/data/dbhandler`).

Co je příčinou tohoto delegování? Vzhledem k tomu, že aplikace implementuje různé databázové třídy a může tím pádem databázi dotazovat prostřednictvím různých knihoven (viz sekci 2.5.2), jsou různé třídy k dispozici i pro čtení dat. Právě tyto třídy implementují rozhraní `IDbHandler` (resp. specifičtější `DbHandlerDb`). Patří mezi ně:

- `DbHandlerMySQLi` (resp. podtřídy `DbHandlerMySQLiResult` a `DbHandlerMySQLiStmt`) – čtení dat dotazovaných databázovou třídou `MySQLiConn`
- `DbHandlerODBC` – čtení dat dotazovaných databázovou třídou `ODBCConn`
- `DbHandlerArray` – čtení databázových dat převedených do serializovatelné podoby (nutné před uložením dat do session).

Třída podporuje zahození nepotřebných dat před serializací do session (viz funkci `DataGetDatabase::add_useless_rows()`).

- `DataGetSession` (implementuje `IDataGet`) – jednoduché čtení sady dat získaných ze session.
- `DataSetDatabase` (implementuje `IDataSet`) – jednoduchý zápis (příp. úprava) datové položky v databázi.
- `DataSetSession` (implementuje `IDataSet`) – jednoduchý zápis (příp. úprava) datové položky v session.
- `DataSetDatabaseMulti` – hromadný zápis sady datových položek do databáze.
- `DataGetText` (implementuje `IDataGetMulti`) – získávání dat importovaných z textového souboru.
- `DataGetRIV` (implementuje `IDataGetMulti`) – získávání dat z **DBF** exportu externí databáze RIV (IS VaVaI).

  Třída provádí abstrakci nad prostou **DBF** tabulkou tak, aby data mohla být získávána v souladu s obvyklou strukturou dat Agendy vědeckých výsledků. Pro záznamy tedy nežádáme jen hodnoty určených atributů (**DBF** sloupců), nýbrž i hodnoty atributů jednotlivých titulů a autorů příslušných danému záznamu, což právě vyžaduje zmíněnou abstrakci.
- `DataGetOBD` (implementuje `IDataGetMulti`) – získávání dat z **XML** exportu externí databáze OBD UPOL.

- `DataExport` (implementuje `IDataGetMulti`) – komplexní čtení dat získaných z databáze. Specializované funkce `DataExport::get_title()` a `DataExport::get_author()` umožňují rychlé vyžádání dat s v určeném formátu.

- `DatabaseSearch` – vyhledání databázových záznamů vyhovujících zadaným podmínkám. Vráceny jsou identifikátory nalezených záznamů.

2.5.4 Filtrace dat

Validita vstupních dat se v aplikaci kontroluje na různých místech. Kontrola správnosti může zahrnovat i úpravy hodnoty, a proto v některých přípech mluvíme o tzv. normalizaci hodnoty. Zásady validace a normalizace definují statické funkce abstraktní třídy `FilterHelp`. Výběr konkrétní funkce závisí na větvení ve funkci `FilterHelp::prepare_args()`.

Samotné provedení filtrace zajišťují podtřídy `FilterHelp`, přizpůsobené konkrétnímu případu použití:

- `FilterVarHelp` – obecná filtrace zadané hodnoty.

- `FilterInsertHelp` – filtrace hodnot při interaktivním vkládání nebo modifikaci položy (třída `InsertScript`).

- `FilterImportHelp` – filtrace hodnot importovaných z textového souboru nebo z externí databáze. (Výsledkem filtrace je objekt třídy `NormVal`.)

- `FilterSearchHelp` – filtrace parametrů vyhledávání.

Zmiňovaná třída `NormVal` představuje pokročilou reprezentaci normalizované hodnoty, obsahující zároveň i původní hodnotu před normalizací, informaci o případné chybě normalizace aj.

Filtrované hodnoty pro sadu importních záznamů shromažďuje třída `FilterImport`, implementující `IDataGetMulti`. Členské funkce `FilterImport::hack_*()` umožňují dodatečné automatické doplnění vybraných importních hodnot.

Filtrované parametry vyhledávání (a také třídící kritéria) shromažďuje třída `FilterSearch`.

2.5.5 Zobrazování a interaktivní vkládání dat

Třídy popsané v této sekci pokrývají funkcionalitu základních případů užití aplikace. Obecnost tříd dovoluje jejich použití pro práci s různými databázovými entitami.

- `ResultsPage` – generování tabulky s hromadným výpisem záznamů, titulů nebo autorů.

- `ResultsFilterOrderPage` (rozšiřuje `ResultsPage`) – generování tabulky s hromadným výpisem záznamů obohacené o podporu vyhledávání, třídění a stránkování. (Parametry vyhledávání a třídící kritéria nese objekt třídy `FilterSearch`.)
- `DetailPage` – generování tabulky s detailním výpisem záznamu, titulu nebo autora.
- `Insert` – abstraktní třída se dvěma podtřídami:
  - `InsertPage` – generování formuláře pro vložení nebo editaci záznamu, titulu nebo autora.
  - `InsertScript` – provedení vložení nebo modifikace záznamu, titulu nebo autora.
- `SearchPage` – generování formuláře pro vyhledávání v údajích záznamu, titulu nebo autora a generování skrytého formuláře pro třídící kritéria.

Kontroly oprávnění před provedením požadované operace provádí ve třídách obvykle členská funkce `::check_up()`.

2.5.6 Porovnání s externími databázemi

Získání dat Před provedením porovnání databází je nutné vyhledat a získat příslušná data z externí a vlastní databáze. K tomu slouží třídy implementující rozhraní `IFetchable`. Funkce rozhraní `IFetchable::get_data_import()` vrací objekt `FilterImport` se sadou dat z externí databáze a funkce `IFetchable::get_ids_export()` vrací pole identifikátorů záznamů z vlastní databáze, vyhledaných ekvivalentním způsobem jako data v externí databázi.

Implementující třídy (`FetchRIV` pro externí databázi RIV a `FetchOBD` pro ext. databázi OBD UPOL) musí vykonat následující činnosti:

- zpracování vyhledávacích parametrů určujících sadu dat externí databáze k provedení porovnávní
- vytvoření požadavku na vyhledání dat v externí databázi a stažení příslušného exportu (kritická část z pohledu údržby – viz sekce 4.4.1 a 4.4.2)
- vyhledání dat ve vlastní databázi ekvivalentním způsobem, jakým byla vyhledána data v externí databázi

Společnou funkcionalitu pojednávaných tříd definuje jejich rodičovská abstraktní třída `FetchCompare`.

Porovnání Data získaná s pomocí funkcí rozhraní `IFetchable` se předávají objektu třídy `Pair`. Ten provede samotné porovnání databází. Jedná se o několik procesů takzvaného *párování* (čili hledání shodného protějška položky v příbuzné databázi[14]):

- párování záznamů získaných z externí databáze se všemi záznamy vlastní databáze (`Pair::pair_records()`)
- párování titulů u záznamů spárovaných v procesu párování záznamů (`Pair::pair_titles()`)
- párování autorů záznamů z externí databáze s autory ve Správě autorů, prováděné pouze pro záznamy nespárované v rámci procesu párování záznamů (`Pair::pair_authors()`)

Porovnání hodnot položek externí a vlastní databáze se při párování provádí způsobem definovaným ve funkci `Pair::compare_pair()`.

Objekt `Pair` je po provedení porovnání uložen i s daty externí a vlastní databáze do session, kde zůstává k dispozici pro výpisy a finální provedení importu (viz níže). Za účelem zvýšení efektivity proto před serializací do session dochází k zahození nepotřebných dat (viz funkce `Pair::clean_data_agenda()`, `Pair::clean_db_authors()`).

Přehledový výpis Zobrazení přehledových tabulek s možností konfigurace importu zajišťují třídy analogické k některým třídám zmiňovaným v sekci 2.5.5. Obecnost tříd dovoluje jejich použití pro práci s různými entitami dat externí i vlastní databáze. Jako zdroj informací pro výpis slouží vždy instance objektu `Pair`.

- `ImportResultsPage` – generování tabulky s hromadným výpisem záznamů, titulů nebo autorů z externí nebo vlastní databáze.
- `ImportDetailPage` – generování tabulky s detailním výpisem záznamu, titulu nebo autora z externí nebo vlastní databáze.

Provedení importu Import vybraných dat z externí databáze zajišťuje třída `ImportScript`. Jako zdroj dat slouží v ní obsažená instance třídy `Pair`. Kontroly oprávnění před provedením importu provádí funkce `ImportScript::check_up()`.

Relevantní skripty Dosavadní popis porovnání s externími databázemi se věnoval PHP třídám, které také zapouzdřují většinu funkcionality. Aby byl proces porovnání co nejjasnější, uvádíme v tabulce 2 i chronologický přehled PHP skriptů, v nichž se operace definované ve třídách volají.

Jak je vidět, některé činnosti vyžadují použití skriptu specifického pro danou externí databázi, jiné vystačí se společným skriptem. Rozdělení přehledového

[14]Pro více informací viz příslušné sekce manuálu pro uživatele a manuálu pro administrátora.

Databáze	Příprava	Zpracování	Výpis	Import
RIV	compare_riv.php	compare_riv_script.php	import_compare.php, import_ajax.php	import_script.php
OBD	compare_obd.php	compare_obd_script.php	import_compare.php, import_ajax.php	import_script.php

Tabulka 2: PHP skripty týkající se porovnání s externími databázemi

výpisu nalezených záznamů na základní stránku (`import_compare.php`) a dodatečně asynchronně doručovaný detail jednotlivého záznamu (`import_ajax.php`) výrazně redukuje množství přenesených dat.

2.5.7 Exportování

Pro export záznamů slouží třídy implementující rozhraní `IExporting`:

- `ExportBiblio` – abstraktní třída pro funkcionalitu bibliografického exportu. Tvar výstupu určují dědící třídy:
  - `ExportBiblioBibtex` – bibliografický export ve formátu seznamu literatury pro BibTEX (viz sekci 1.5.2).
  - `ExportBiblioDBLP` – bibliografický export ve formátu *DBLP XML* (viz sekci 1.5.3).
- `ExportCSV` – export dat do CSV tabulky.
- `ExportText` – export dat do textového souboru (více v sekci 2.5.8).

2.5.8 Import a export do textového souboru

Před započetím importu nebo exportu je nezbytné zpracovat šablonu textového souboru, což zajišťuje abstraktní třída `Text`. Její statická funkce `Text::template_example()` navíc pro zadanou šablonu generuje vzor importního souboru.

Rozšiřující třídy `FetchText` a `ExportText` využívají šablonu načtenou v rodičovské třídě a zajišťují zpracování importního, resp. vytvoření exportního textového souboru.

Koncepce importu záznamů z textového souboru představuje po technické stránce zvláštní případ porovnání s externí databází (viz sekci 2.5.6). Třída pro získání importních dat `FetchText` implementuje (stejně jako třídy `FetchRIV` a `FetchOBD`) rozhraní `IFetchable`. Její funkce `FetchText::get_ids_export()` namísto identifikátorů záznamů vlastní databáze vrací hodnotu logické nepravdy, protože čtení ani výpis dat vlastní databáze se neprovádí. Oproti tomu `FetchText::get_data_import()` vrací klasicky sadu importních dat, která lze později předat instanci třídy `Pair`.

`Pair` obsahující data importovaná z textového souboru provede stejné činnosti jako při porovnání databází s tím rozdílem, že importní záznamy nepáruje se

záznamy vlastní databáze, nýbrž ponechá všechny nespárované. Pro výpisy a finální provedení importu se pak použijí stejné třídy jako při porovnání databází (viz sekci 2.5.6).

Relevantní skripty Podobně jako v technickém popisu porovnání s externí databází znázorníme i nyní proces importu z pohledu PHP skriptů. Přehled obsahuje tab. 3. (Při srovnání s tab. 2 si lze všimnout použití několika společných skriptů.)

Příprava	**Zpracování**	**Výpis**	**Import**
insert_file.php	insert_file_script.php	import_text.php, import_ajax.php	import_script.php

Tabulka 3: PHP skripty týkající se importu z textového souboru

2.5.9 Pomocné třídy

Třída `DbUpdate` provádí aktualizaci databázového schematu a pomáhá s výpisy.

Třídy ve složce `class/lib` zajišťují napříč aplikací specifické služby. Seznam tříd:

- `Connection` – systémová třída (viz sekci 2.5.2).
- `Columns` – překládá interní identifikátory (viz sekci 2.2.2) na reálné názvy databázových atributů. Instance třídy je dostupná v globální proměnné `$COLS`.
- `ColumnsPrefix` – jako `Columns`, ale k přeloženému názvu přidává prefix identifikující databázovou tabulku (vrácený název atributu s prefixem má tedy tvar „PREFIX___nazev_atributu“). Instance třídy je dostupná v globální proměnné `$COLS_P`.
- `UsersColumns` – jako `Columns`, ale pracuje s atributy tabulky uživatelů. Instance třídy je dostupná v globální proměnné `$UCOLS`.
- `SessionHandlerAdjusted` – třída rozšiřuje nativní třídu `SessionHandler` a slouží tak k přizpůsobení práce se session. V našem případě provádí pouze mazání nepotřebných souborů (exporty z externí databáze).
- `ExceptionAdjusted` – třída explicitně volaných výjimek. Rozšiřuje nativní třídu `Exception`. Obvyklou obsluhu výjimek (přesměrování na stránku se zprávou o chybě) definuje globální funkce `handle_exception()`.
- `Redirect` – třída zajišťující přesměrování.
- `Status` – obsahuje seznam chybových a informativních zpráv a zajišťuje různé druhy jejich výpisu. (Nejčastěji se zprávy zobrazují na stránce `info.php`.)

- `HeaderFooter` – generování hlavičky a patičky HTML stránky.

2.5.10 Implementační aspekty

TableTrait *Trait* představuje v jazyce PHP speciální druh třídy, která je určena ke sdílení funkcionality s dalšími třídami. V kontrastu k zásadám dědičnosti, kdy třída může používat funkcionalitu nanejvýš jednoho rodiče, je sdílení funkcionality *traitů* povoleno u jediné třídy pro více *traitů* současně. Více informací obsahuje [7].

Funkcionalitu *traitu* `TableTrait` používají v Agendě věd. výsledků třídy, které provádí nějakou činnost pro jednotlivé atributy tabulek databázového schematu. Databázové schema má pevně danou strukturu a od té se odvíjí i architektura těchto tříd. `TableTrait` deklaruje abstraktní funkce, kterými tuto architekturu ve třídách vynucuje:

- go_col_id() – provedení činnosti pro atribut identifikátor (*i_id*)
- go_col_username() – provedení činnosti pro atribut vlastník (*i_username*)
- go_col_public() – provedení činnosti pro atribut veřejnost (*i_public*)
- go_col_REST() – provedení činnosti pro variabilní atributy
- go_col_title(), go_col_author() – provedení činnosti pro tituly a autory záznamu (funkce neodpovídá konkrétnímu atributu a náplň její činnosti se u různých tříd liší)

Funkce se používají pro daný typ atributu napříč tabulkami. Zmiňovanou činností, kterou funkce pro daný atribut provádějí, může být generování příslušné části HTML tabulky s výpisem (třídy `ResultsPage`, `DetailPage`, `InsertPage`, `SearchPage` aj.), filtrace hodnoty atributu (třídy `FilterImport`, `FilterSearch`) nebo příprava hodnoty k zápisu (třída `InsertScript`).

Použití generátorů *Generátor* je v jazyce PHP iterační objekt vytvořený jednoduchým způsobem z obvyklé funkce, která však namísto klasického `return` používá pro vracení hodnot klíčové slovo `yield`. Pro více informací viz [7].

V Agendě věd. výsledků se generátory užívají významným způsobem ve třídách pro hromadný výpis položek (`ResultsPage` a `ImportResultsPage`). Vzhledem k tomu, že tabulka s výpisem se tiskne po řádcích, členské funkce `::go_col_*()`, vypisující jednotlivé sloupce, by se měly volat opakovaně pro každý řádek tabulky (tj. pro každou položku výpisu). Použití principu generátorů však dovoluje zavolat každou `::go_col_*()` funkci jen na začátku výpisu a z vráceného iteračního objektu pak vyžadovat další výpisy, které generuje její vnitřní cyklus. Zvýšení efektivity spočívá v tom, že vnitřní cyklus neobsahuje inicializaci proměnných společných pro všechny výpisy – tato inicializace se provede pouze při prvním volání funkce.

2.5.11 Práce se souborovým systémem

Aplikace vytváří soubory v souvislosti se získáváním exportů z externích databází. K zápisu jsou určeny složky uvedené v direktivách *files_import_riv* a *files_import_obd* konfiguračního souboru `src/conf/application.ini.php`. Soubory jsou automaticky mazány, a to buď ihned po zpracování dat, nebo dodatečně při provádění úklidu (funkce `SessionHandlerAdjusted::gc()`).

Dočasné uložení importních textových souborů zaslaných uživatelem neobstarává aplikace, nýbrž přímo PHP (srv. direktivu *upload_tmp_dir*).

3 Manuál pro uživatele

3.1 Požadavky na prohlížeč

Aplikace vyžaduje podporu JavaScriptu a zasílání cookies. Při vypnutém JavaScriptu se zobrazí varovná hláška.

Aplikace byla testována pro prohlížeče Internet Explorer 11, Edge 25, Mozilla Firefox 46 a Google Chrome 50.

3.2 Přihlášení a správa účtu

Pro přihlašování a činnosti spjaté s uživatelským účtem je vyhrazena pravá část lišty s nabídkou. Po úspěšném přihlášení lze příkazem *Upravit účet* změnit heslo nebo kontaktní email. Po určité době nečinnosti může dojít k automatickému odhlášení.

Založení nového nebo smazání existujícího uživatelského účtu, stejně jako obnovení zapomenutého hesla je oprávněn provést pouze administrátor aplikace.

V nepřihlášeném stavu aplikace umožňuje pouze prohlížení, vyhledání a export veřejných záznamů, s omezením na jejich veřejné údaje.

3.3 Titulní stránka s výpisem záznamů

Na titulní stránce se ve výchozím stavu zobrazují všechny záznamy o vědeckých výsledcích v databázi. V tabulce jsou pro každý záznam uvedeny nejdůležitější údaje, podrobnější detail záznamu lze zobrazit kliknutím na ikonku lupy. Neveřejné záznamy a neveřejné údaje veřejných záznamů jsou v nepřihlášeném stavu ze všech výpisů vyřazeny.

Sloupec *Název záznamu* zobrazuje název záznamu v jeho původním jazyce, sloupec *Autoři záznamu* pak výpis autorů typu *Autor*[15]. Výpis autorů zachovává pořadí specifikované při vkládání záznamu. Druh záznamu se kvůli úspoře místa vypisuje jako písmenný kód, jeho význam odhaluje až popisek zobrazený po najetí myši. Ve sloupci URL se zobrazuje vždy nejvýše jeden odkaz na online verzi záznamu; případné další odkazy se vypisují v detailu záznamu.

[15] Autoři typu *Editor* nejsou na titulní stránce zobrazováni.

Tabulka se záznamy používá stránkování. Další stránku výpisu záznamů lze zobrazit poklikem na číslo stránky v ovládacím prvku nad tabulkou. Informace o počtu položek se přitom týká všech nalezených položek, nikoli pouze položek na aktuální stránce.

Třídění tabulky se záznamy

Tabulku lze třídit podle kritérií odpovídajících sloupcům tabulky – nové třídící kritérium se přidává stiskem šipky nad popiskem příslušného sloupce. Aplikace podporuje vícekriteriální třídění, které nejprve záznamy setřídí podle údaje odpovídajícího nejvýznamnějšímu kritériu, záznamy, jež mají tento údaj shodný, pak znovu setřídí podle údaje odpovídajícího druhému nejvýznamnějšímu kritériu, a tak opakovaně až do posledního třídícího kritéria. Všechna použitá třídící kritéria jsou vypsána ve štítcích nad tabulkou.

Každé nově přidané třídící kritérium se považuje za aktuálně nejvýznamnější a je předřazeno stávajícím třídícím kritériím[16]. Počet současně použitých třídících kritérií je omezen a při překročení limitu dojde v seznamu kritérií k vyřazení toho momentálně nejméně významného. Obvykle jsou už ve výchozím stavu k dispozici předvolená třídící kritéria.

Většina údajů se třídí očekávatelným způsobem (čísla a data numericky, řetězce abecedně). Výčtové údaje se třídí abecedně, anebo specifickým způsobem podle důležitosti. Ve druhém z případů bývá u třídících šipek umístěna hvězdička, zobrazující po najetí myši specifické pořadí třídění.

Hromadná manipulace se záznamy

Tabulka se záznamy obsahuje v levé části zaškrtávací pole pro každý zobrazený záznam. Se vybranými záznamy lze provádět hromadné operace, reprezentované tlačítky pod titulkem *Vybrané záznamy*. V nepřihlášeném stavu je k dispozici pouze možnost exportu. Pro vybrání všech záznamů na stránce slouží zaškrtávací políčko v záhlaví tabulky se záznamy. Pro export všech nalezených záznamů (tj. ne pouze těch na aktuální stránce) lze použít tlačítko *Exportovat vše*.

Vyhledávání

Kompletní výpis záznamů zobrazovaný ve výchozím stavu lze zúžit specifikováním vyhledávacích parametrů.

Formulář pro vyhledávání se zobrazí po stisku tlačítka *Vyhledávání*. Výchozí je základní verze formuláře, umožňující vyhledávat podle nejfrekventovanějších údajů. Další vyhledávací parametry se zpřístupní po kliknutí na tlačítko *Rozšířené vyhledávání*. Rozšířená varianta umožňuje pro číselné údaje a data zadat

[16] Chceme-li tedy např. třídit záznamy podle roku a záznamy se stejnou hodnotou roku dodatečně setřídit podle jejich druhu, musíme přidat nejprve třídící kritérium pro druh a poté teprve pro rok, tzn. kritéria přidáváme od nejméně významného.

namísto přesné hodnoty rozsah hodnot. Rozsah může být zleva nebo zprava otevřený. Při přechodu mezi základní a rozšířenou variantou vyhledávání dochází ke smazání dosavadních vyhledávacích parametrů.

V případě vyplnění více vyhledávacích parametrů uvedených pod titulkem (na záložce) *Titul záznamu* budou hledány záznamy obsahující nějaký titul, jenž odpovídá všem těmto vyhl. parametrům *současně*. Přitom titul je struktura nesoucí název, abstrakt a klíčová slova záznamu v jednom konkrétním jazyce (viz sekci 2.2.1). Z toho plyne, že nemá smysl zadávat pro *Titul záznamu* vyhledávací parametry v různých jazycích (titul, který by vyhovoval všem současně, zpravidla neexistuje).

Analogicky, při vyplnění více vyhledávacích parametrů pod titulkem (na záložce) *Autor záznamu* se hledají záznamy s nějakým autorem, jenž splňuje všechny parametry současně.

Pravá část vyhledávacího formuláře obsahuje ikonky, jež po najetí myši informují o způsobu vyhledání. V případě vyhledávání podle údaje Druh záznamu se pro hodnotu „J – článek“ hledají i všechny poddruhy článku (Jimp, JSC, Jrec a Jneimp).

Možnosti přihlášeného uživatele

Po přihlášení se pro každý záznam, jehož je přihlášený uživatel vlastníkem, zpřístupní možnost editace a smazání (ikonky tužky a křížku v tabulce se záznamy). Vlastněné záznamy lze mazat také hromadně příslušným tlačítkem pod titulkem *Vybrané záznamy*.

3.4 Detail záznamu

Detail záznamu lze zobrazit kliknutm na ikonku lupy u příslušného záznamu v tabulce na titulní stránce.

Údaje o záznamu se člení do tří záložek, přičemž záložka *Přehled* obsahuje všechny informace vyjma seznamů autorů a titulů, které jsou dostupné na samostatných záložkách. Pro každý titul na záložce *Tituly záznamu* a pro každého autora na záložce *Autoři záznamu* lze kliknutím na ikonku lupy zobrazit jeho detail. Po najetí myši na popisek kteréhokoli údaje se zobrazí jeho bližší popis.

Všechny neveřejné údaje jsou v nepřihlášeném stavu vyřazeny z výpisu.

3.5 Možnosti exportu

Exportovat lze záznamy z tabulky záznamů na titulní stránce, a také záznamy nalezené ve vlastní databázi při porovnání s externí databází. Typ exportu vždy určuje rozbalovací seznam. V nepřihlášeném stavu jsou z exportů vyřazeny neveřejné údaje.

Při exportu do **seznamů literatury** závisí skladba exportovaných údajů na druhu exportovaného záznamu. Výjimku tvoří export pro vlastní LaTeX, konfigurovaný uživatelsky definovanou šablonou. Šablonu lze editovat po kliknutí

na odkaz *Upravit šablonu*, který se zobrazí po výběru příslušného exportu ze seznamu exportů. Pokyny pro definici šablony jsou popsány přímo v rozhraní aplikace. Změněná definice šablony zůstává uložena do uzavření prohlížeče nebo do odhlášení.

Export do **textového souboru** se provádí na základě některé z šablon definovaných administrátorem aplikace. Některé šablony mohou být dostupné pouze v přihlášeném stavu. S využitím totožných šablon se provádí také import záznamů z textového souboru (viz sekci 3.9).

CSV export se snaží poskytnout co nejkomplexnější sadu informací. To není jednoduché zejména pro údaje týkající se autorů a titulů záznamu. Autoři záznamu jsou zřetězeni v jediném sloupci „Autor“ a z prostorových důvodů dochází k vynechání některých jejich údajů. Každému údaji týkajícímu se titulu záznamu pak přísluší vlastní sloupec s výpisem hodnot pro tituly záznamu v různých jazycích. Všechny ostatní údaje jsou uloženy ve vlastním sloupci běžným způsobem.

3.6 Správa autorů

Základ databázových dat představují autoři, které lze přiřazovat jednotlivým záznamům. Vytváření, editaci a mazání těchto autorů zajišťuje rozhraní na stránce *Správa autorů*. Před vložením nového záznamu je nutné, aby všichni jeho autoři již existovali ve Správě autorů.

Po vložení nového autora do *Správy autorů* je mu za vlastníka přidělen uživatel, který vkládání provedl. Výhradně tento uživatel a *administrátor* pak mohou autora editovat nebo smazat (ikonky tužky a křížku v tabulce s výpisem autorů). Změny provedené v rámci editace se projeví ve všech záznamech, kde byl autor použit. Stejnětak smazání autora odstraní jeho výskyty ve všech záznamech; z toho důvodu lze mazat pouze autory, kteří nebyli dosud v žádných záznamech použiti. K mazání jakýchkoli autorů má nicméně oprávnění *administrátor*.

Neprovádí se žádné kontroly unikátnosti vkládaných autorů – v databázi lze evidovat několik autorů se zcela totožnými údaji. Jednoznačné určení autora je tedy možné pouze prostřednictvím identifikátoru.

Mimo popsané rozhraní lze nové autory do Správu autorů přidat pouze při importu z textového souboru nebo během porovnání s externí databází, a to v dialogu *Vytvoření autorů v Agendě* (viz sekci 3.9.2, část o vytvoření autorů před importem).

3.7 Vložení nového záznamu

Proces vložení nového záznamu zahajujeme volbou z nabídky *Vložit záznamy – Ručně (rozpracovaný záznam)*. Tzv. *rozpracovaný záznam* označuje nekompletní záznam, uložený dočasně v paměti aplikace (záznam se smaže až při vložení záznamu do databáze nebo při odhlášení). Vyplňování rozpracovaného záznamu lze tedy přerušit, přejít na chvíli do jiné části aplikace, a opět se k němu vrátit.

Samotné vložení záznamu do databáze se provádí až na závěr stiskem tlačítka *Vložit záznam* v pravém horním rohu.

Struktura rozhraní pro vkládání záznamu je podobná rozhraní pro detail záznamu (viz sekci 3.4). Záložka *Přehled* umožňuje vyplnit všechny údaje rozpracovaného záznamu vyjma titulů a autorů. Bezpodmínečně nutné je vyplnění polí označených hvězdičkou. Pole s červeným ohraničením se obvykle vyplňují pro aktuálně zvolený druh záznamu, ale do databáze lze vložit i prázdnou hodnotu. Žlutě zvýrazněná pole si zaslouží pozornost proto, že jejich obsah se zobrazuje v tabulce s výpisem záznamů na titulní stránce. Tlačítko *Uložit změny* zapíše rozpracovaný obsah polí do paměti aplikace (nikoli do databáze).

Prostřednictvím záložky *Tituly záznamu* lze přidávat, editovat a mazat tituly rozpracovaného záznamu. Změny se zapíší do paměti aplikace (nikoli do databáze) ihned po provedení operace. Záznam mívá obvykle několik titulů různého jazyka, ale nastavení stejného jazyka pro více titulů současně je taky přípustné. Záznam by měl obsahovat alespoň titul v jazyce shodném s původním jazykem záznamu (viz údaj o původním jazyce na záložce *Přehled*), protože ten se přednostně používá při výpisech, exportech a párování záznamů během procesu porovnání s externí databází. Přípustná je nicméně i varianta záznamu bez jakéhokoli titulu.

Záložka *Autoři záznamu* pracuje s autory záznamu podobným způsobem, jako záložka *Tituly záznamu* s tituly. Rozdíl nastává při vkládání a editaci autora, kdy namísto ručního vyplňování údajů pouze vybereme z nabídky již existujících autorů v databázi a údaje zvoleného autora se automaticky doplní. (V případě, že autor není v nabídce, je nutné jej nejprve vytvořit ve Správě autorů – viz sekci 3.6.) To se týká údajů spjatých s autorem samotným[17]. Kromě toho je nutné ručně vyplnit údaje charakterizující vztah autora k aktuálnímu záznamu (údaje zahraniční, stát a afiliace se přitom automaticky předvyplní podle výchozích hodnot aktuálně vybraného autora). Mezi nejdůležitější údaje patří typ autora a pořadí[18], ovlivňující zejména způsob výpisu[19]. Jednomu záznamu nelze přiřadit více autorů představujících tutéž osobu (tj. odkazujících na stejného autora ve Správě autorů)[20]. Varianta záznamu bez jakýchkoli autorů je přípustná.

Po najetí myši na popisek vyplňovaného údaje se zobrazí jeho bližší popis. Přímá nápověda pro vyplnění pak bývá uvedena na pozadí vstupního pole.

Nově vloženému záznamu je jako vlastník přidělen aktuální uživatel. Výhradně tento uživatel a *administrátor* mohou záznam editovat nebo smazat.

Neprovádí se žádné kontroly unikátnosti vkládaných záznamů – v databázi lze evidovat několik záznamů se zcela totožnými údaji. Jednoznačné určení záznamu je tedy možné pouze prostřednictvím identifikátoru.

[17] Tyto údaje jsou vždy zvýrazněny modrým pozadím.

[18] V praxi se obvykle používá pořadí sestavené podle důležitosti autorů nebo abecední pořadí.

[19] Nejprve se uvádí autoři typu *Autor* podle určeného pořadí, poté stejným způsobem autoři typu *Editor*. V některých případech, jako např. v tabulce se záznamy na titulní stránce, se autoři typu *Editor* neuvádí vůbec.

[20] To platí i v případě, jsou-li autoři odlišného typu – jeden typu *Autor* a druhý typu *Editor*.

3.8 Editace záznamu

Stránka s editací záznamu se zobrazí po kliknutí na ikonku tužky u příslušného záznamu v tabulce na titulní stránce.

Uživatelské rozhraní je totožné s rozhraním pro vkládání záznamu (viz sekci 3.7) s tím rozdílem, že veškeré změny se již zapisují přímo do databáze.

3.9 Import z textového souboru

Hromadně lze data do aplikace importovat zasláním textového souboru s předem určenou strukturou.

3.9.1 Příprava importu

Rozhraní pro zadání importního textového souboru a šablony popisující jeho strukturu se zobrazí volbou z nabídky *Vložit záznamy – Importem z textového souboru*. Pro import z textového souboru jsou k dispozici stejné šablony jako pro export (srv. sekci 3.5). Při použití stejných šablon platí až na výjimky dualita importu a exportu – exportovaná data lze znovu importovat, a to při zachování jejich původního významu. Pro podrobnosti viz soubor `doc/textfile.odt`.

Zaškrtávacími políčky v části *Možnosti importu* lze povolit automatické doplnění některých importních hodnot pro případ, že by byly neplatné (prázdné nebo nevalidní). Více informací lze získat po najetí myši na ikonku s otazníkem.

Po stisku tlačítka *Zahájit import* a úspěšném odeslání importního souboru se na serveru spustí časově a výpočetně náročný proces zpracování importních dat. V případě vyčerpání systémových zdrojů může dojít k zastavení programu a zobrazení chybové hlášky.

Struktura textového souboru Požadovanou strukturu importního textového souboru zachycuje vzorový soubor generovaný pro příslušnou šablonu kliknutím na odkaz *Generovat vzor importního souboru*. Řetězce uzavřené ve složených závorkách jsou ve vzoru určeny k nahrazení za importní hodnoty. Je nezbytné vyplnit všechny hodnoty, vyjma těch označených za nepovinné. Texty následující za znakem „#“ představují komentáře a před importem musí být odstraněny. Mimo proměnlivé importní hodnoty musí textový soubor obsahovat všechny řetězce tak, jak je uvedeno ve vzoru. V případě neshody importního souboru a šablony se v importním souboru buď přejde na další řádek, anebo vznikne chyba a pokračuje se dalším záznamem. Jednotlivé záznamy se oddělují speciálním řetězcem (tzv. *oddělovač*), uvedeným na posledním řádku vzoru. Závěr importního souboru vždy obsahuje jedno nebo více odřádkování; řídit se lze podle počtu odřádkování před oddělovačem ve vzoru. Přestože jde o závěr souboru, za odřádkováními může ještě následovat oddělovač.

Pokud se některý údaj pro danou šablonu v importním textovém souboru vůbec neuvádí, jako importní hodnota údaje se použije neurčená hodnota (prázdný řetězec), je-li údaj výčtového typu, a prázdná hodnota pro údaje ostatních typů.

Pro detailní informace o struktuře importního souboru je nutné kontaktovat administrátora aplikace. Bližší představu lze získat také prohlédnutím exportů provedených dle šablony zamýšlené pro import. Podrobnosti o zpracování importního souboru obsahuje dokumentace `doc/textfile.odt`.

Předpokládá se, že importní soubor používá kódování UTF-8.

3.9.2 Přehled importních záznamů

Po úspěšném zpracování importních dat se otevře rozhraní umožňující jejich inspekci a kofiguraci importu. Všechna importní data jsou uložena v paměti aplikace, a to až do provedení importu nebo do odhlášení. K dosud neimportovaným datům se proto lze vrátit – stačí znovu zobrazit rozhraní pro přípravu importu (volba nabídky *Vložit záznamy – Importem z textového souboru*) a přejít na odkaz *Vrátit se k datům předchozího importu*.

Importní záznamy se v přehledovém výpisu dělí do dvou skupin:

- *Validní importní záznamy* – záznamy připuštěné k importu (tlačítko *Importovat vybrané*)
- *Chybné importní záznamy* – záznamy, jejichž čtení nebylo možné dokončit kvůli nesouladu s importní šablonou (nesoulad struktury záznamu, nevyplněná povinná hodnota). Pro každý záznam je uvedeno číslo řádku, na kterém v importním textovém souboru chyba vznikla.

Každá hodnota obsažená v importním souboru prochází procesem normalizace, jenž zahrnuje kontrolu její validity a případné provedení úprav, jako jsou např. ořezání bílých znaků nebo zalomení textu víceřádkové hodnoty. V případě selhání normalizace se pro import použije výchozí nebo automaticky doplněná hodnota (podle nastavení při přípravě importu).

Novým záznamům vloženým při importu je za vlastníka přidělen aktuální uživatel. Neprovádí se žádné kontroly unikátnosti vkládaných záznamů mezi sebou navzájem, ani vzhledem k záznamům v databázi. Neuvážený import proto může vést ke vzniku duplicit.

Pro hromadné i detailní zobrazení importních hodnot platí, že se vždy zobrazují v původní podobě – nedochází k převedení na uživatelský způsob zobrazení (např. ikona s odkazem pro údaj o URL), výčtové hodnoty se zobrazují bez uživatelských popisků a do výpisu jsou zahrnuty i po sobě následující bílé znaky a odřádkování.

Vedle importní hodnoty nebo na jejím místě mohou být v hromadném i detailních výpisu zobrazeny informativní ikonky. Patří mezi ně:

- ikona vykřičník (normalizace hodnoty selhala; bližší popis se zobrazí po najetí myší)
- ikona plus (byla použita automaticky doplněná hodnota – podle nastavení v přípravě importu)

- ikona minus (hodnota je prázdná)
- ikona otazník (výčtová hodnota má význam neurčené hodnoty)
- ikona „i“ (normalizovaná hodnota je stejná jako původní importní hodnota)

Hromadný výpis validních záznamů Tabulka hromadného výpisu validních záznamů je uspořádána analogicky k tabulce se záznamy na titulní stránce. Zobrazují se normalizované, nikoli původní importní hodnoty. Záznam lze vyřadit nebo přidat do importu pomocí zaškrtávacího políčka v levé části tabulky, políčko v záhlaví pak hromadně vybere všechny záznamy. Při selhání normalizace nebo automatickém doplnění kterékoli hodnoty v rámci záznamu se v levé části tabulky zobrazí globální informativní ikonka příslušného typu (viz výpis ikon výše). Na záznamy, jejichž některý autor byl spárován s více než jedním protějškem ve Správě autorů (párování autorů viz níže v sekci o detailu záznamu), upozorní ikonka seznamu zobrazená na stejném místě.

Detail záznamu se vysouvá kliknutím na tlačítko *Podrobnosti*.

Celkový počet aktuálně vybraných záznamů k importu udává číslo uvnitř tlačítka *Importovat vybrané*.

Detail záznamu Detailní zobrazení validního záznamu, skryté pod tlačítkem *Podrobnosti*, je uspořádáno do tří záložkek, analogicky k detailu běžného záznamu na titulní stránce. Pro každý údaj se zobrazují dvě hodnoty – *Původní importní hodnota* a *Normalizovaná importní hodnota*. Pro import je podstatná normalizovaná hodnota.

Na záložce *Tituly záznamu* lze zaškrtávacími políčky *Importovat titul* vyřadit nebo přidat tituly do importu.

Konfigurace importu autorů na záložce *Autoři záznamu* opět umožňuje vyřazení nebo přidání autora do importu pomocí zaškrtávacího políčka *Importovat autora*. Navíc lze pro import nastavit použití v databázi již existujícího autora. Jak bylo popsáno v sekci o vkládání záznamu (3.7), autor, který má být přiřazen nějakému záznamu, musí existovat anebo být předem vytvořen ve Správě autorů (3.6). Každý autor importního záznamu je proto porovnán (tzv. *párován*) se všemi autory ve Správě autorů a nalezení totožní protějšci jsou posléze nabídnuti k použití při importu. (Pro zjištění, co musejí autoři splňovat, aby byli považováni za totožné, viz odkaz *Podrobnosti o párování autorů*.) Alternativou k použití autora existujícího ve Správě autorů je automatické vytvoření nového autora tamtéž, a to přímo v rámci importu.

Pokud byl importní autor spárován alespoň s jedním protějškem ve Správě autorů, použití tohoto protějška při importu se nastaví zaškrtávacím políčkem *Použít spárovaného autora v Agendě*. Pokud byl spárován s více protějšky, protějška žádaného pro import lze vybrat z rozbalovacího seznamu. Údaje aktuálně vybraného protějška se zobrazují ve sloupci *Hodnota spárovaného autora v Agendě*. Je vhodné tyto hodnoty srovnat s hodnotami importního autora (sloupec *Normalizovaná importní hodnota*), protože obecně nemusejí být všechny shodné. Nebyl-li

pro importního autora ve Správě autorů nalezen žádný totožný protějšek (tzv. nespárovaný importní autor), bude v každém případě ve Správě autorů při importu vytvořen nový autor.

Importní hodnoty použité pro údaje příslušející autorovi obecně (ve výpisu zvýrazněny tmavším pozadím) závisí na tom, zda je pro import použit totožný protějšek autora ve Správě autorů. Pokud ano, hodnoty ve sloupci *Normalizovaná importní hodnota* přestávají být rozhodující, protože při importu se použije existující autor s hodnotami ve sloupci *Hodnota spárovaného autora v Agendě*. Pokud ne, bude ve Správě autorů vytvořen nový autor, a to zcela podle hodnot ve sloupci *Normalizovaná importní hodnota*.

Pro import údajů evidovaných pro autora ve vztahu ke konkrétnímu záznamu (typ, pořadí atd.) nemá (narozdíl od obecných údajů autora zmiňovaných v předchozím odstavci) použití protějška ve Správě autorů žádný význam – vždy budou použity hodnoty ve sloupci *Normalizovaná importní hodnota*.

V rámci jednoho záznamu nelze použít pro import dva stejné spárované autory ve Správě autorů. V případě vytvoření nového autora ve Správě autorů je tomuto autorovi přidělen za vlastníka aktuální uživatel. Žádné kontroly unikátnosti nově vytvářených autorů se neprovádí – neuvážený import proto může vést ke vzniku duplicit.

Ve výchozím stavu je import konfigurován jako úplný – importují se všechny záznamy se všemi svými tituly a autory. Pro import každého z autorů je předvolen první totožný protějšek nalezený ve Správě autorů. Nebyl-li žádný protějšek nalezen, ve Správě autorů je při výchozím stavu importu vytvořen nový autor.

Vytvoření autorů před importem Jak bylo popsáno výše, každý autor importního záznamu, který byl zvolen k importu a ke kterému neexistuje anebo nebyl vybrán totožný protějšek ve Správě autorů, bude při importu vytvořen ve Správě autorů jako nový autor. Toto chování je nežádoucí v případě, že několik takto vytvářených autorů představuje totožnou osobu – mezi autory ve Správě autorů by tak došlo ke vzniku duplicit. Předejít tomu lze vytvořením příslušného autora ve Správě autorů ještě před importem, díky čemuž se pak importní autoři představující totožnou osobu spárují s jediným nově vytvořeným protějškem ve Správě autorů a ke vzniku duplicit nedojde.

Autory lze ve Správě autorů vytvořit obvyklým způsobem na stránce *Správa autorů* (viz sekci 3.6), anebo automatizovaně s pomocí dialogu *Vytvoření autorů v Agendě* (tlačítko pro otevření dialogu se zobrazí nad tabulkou s importními záznamy pouze v případě, že mezi importními autory existuje nějaký, který nebyl spárován se žádným protějškem). Změny v párování importních autorů se projeví až po stitknutí tlačítka *Obnovit autory z Agendy*.

Výběr autorů pro vytvoření ve Správě autorů je nutné v dialogu *Vytvoření autorů v Agendě* provést pozorně, tak aby nedošlo k vytvoření duplicitních autorů, kterému se právě snažíme předejít.

Informaci o počtu importních autorů nespárovaných se žádným protějškem ve Správě autorů[21] (a tím pádem i o potenciálním nebezpečí vložení duplicitních autorů při importu) nese pro každý záznam obdélník s číslem zobrazený v levé části tabulky s výpisem záznamů a pro všechny záznamy souhrnně obdélník s číslem nad tabulkou výpisu záznamů. Zobrazené číslo se vždy vztahuje k úplné (tj. výchozí) podobě importu, kdy jsou pro import vybráni všichni dostupní importní autoři.

3.10 Porovnání s externími databázemi

Funkcionalita porovnání s externími databázemi umožňuje srovnat vybraná data z příbuzné externí databáze s daty vlastní databáze a hromadně importovat nalezené rozdíly. Podporováno je porovnání s databázemi RIV (IS VaVaI) (viz sekci 1.4.1) a OBD UPOL (viz sekci 1.4.2), přičemž uživatelské rozhraní porovnání je pro obě varianty stejné.

3.10.1 Příprava porovnání

Před zahájením porovnání je nutné specifikovat vyhledávací parametry, které určí sadu záznamů externí databáze k provedení porovnání. Formulář pro zadání parametrů se zobrazí volbou požadované externí databáze z nabídky *Externí databáze*. Struktura formuláře odpovídá formuláři pro vyhledávání na titulní stránce. Ikonky v pravé části formuláře poskytují po najetí myši důležité informace o tom, jak vyhledávání v externí databázi probíhá.

Zaškrtávacími políčky v části *Možnosti importu* lze povolit automatické doplnění některých importních hodnot pro případ, že by byly neplatné (prázdné nebo nevalidní). Více informací lze získat po najetí myši na ikonku s otazníkem. Automaticky doplněné hodnoty nebudou použity při porovnávání (párování) záznamů externí databáze se záznamy vlastní databáze, ani při importu dat do již existujícího záznamu ve vlastní databázi[22].

Tlačítkem *Zahájit porovnání* se na serveru spustí časově a výpočetně náročný proces, zahrnující stahování exportu z externí databáze, zpracování jeho dat a srovnání těchto dat s daty vlastní databáze. V případě vyčerpání systémových zdrojů může dojít k zastavení programu a zobrazení chybové hlášky. Při dalším pokusu o porovnání databází je pak vhodné zadat specifičtější vyhledávací parametry, které sníží objem zpracovávaných dat.

[21] Číslo nezahrnuje ty importní autory, kteří s nějakým protějškem spárováni byli, ale žádný z těchto protějšků nebyl v konfigurovaci importu použit.

[22] V takovém případě by mohlo dojít k přemazání platných hodnot záznamu ve vlastní databázi automaticky doplněnými (a proto ne příliš cennými) hodnotami záznamu z externí databáze.

3.10.2 Přehled nalezených záznamů

V případě úspěšného stažení a zpracování dat z externí databáze se otevře rozsáhlé rozhraní umožňující inspekci nalezených dat a konfiguraci jejich importu. Všechna importní data jsou uložena v paměti aplikace, a to až do provedení importu nebo do odhlášení. K dosud neimportovaným datům se proto lze vrátit – stačí znovu zobrazit formulář pro vyhledání v externí databázi (volba z nabídky *Externí databáze*) a přejít na odkaz *Vrátit se k datům předchozího porovnání*.

Rozhraní s přehledem nalezených záznamů je členěno do dvou záložek. Záložka *Importní záznamy* obsahuje všechny záznamy nalezené pro zadané vyhledávací parametry v externí databázi. V záložce *Záznamy nalezené v Agendě* je naopak k dispozici výpis všech záznamů vlastní databáze vyhledaných ekvivalentním způsobem jako záznamy z externí databáze. Na záložce *Importní záznamy* volíme a konfigurujeme záznamy k importu (tlačítko *Importovat vybrané*), na záložce *Záznamy nalezené v Agendě* vybíráme záznamy k exportu (tlačítko *Exportovat vybrané*).

Záznamy pro import Každý ze záznamů nalezených v externí databázi byl během zpracování dat porovnáván se záznamy vlastní databáze (tzv. *párování záznamů*). Na základě tohoto srovnání dělíme záznamy na záložce *Importní záznamy* do tří skupin:

- *Spárované importní záznamy* – záznamy z externí databáze, k nimž byl nalezen totožný protějšek (spárovaný záznam) ve vlastní databázi. Import údajů proběhne do tohoto již existujícího protějška. Pro zjištění, co musejí záznamy splňovat, aby byly považovány za totožné, viz odkaz *Podrobnosti o párování*.

  Import rozdílů do záznamu ve vlastní databázi je možný pouze pro údaje záznamu a jeho titulů. Autoři spárovaných záznamů jsou s ohledem na provedené párování záznamů považováni za totožné[23] a import rozdílů není pro jejich údaje podporován.

- *Nespárované importní záznamy* – záznamy z ext. databáze, k nimž žádný totožný protějšek ve vlastní databázi nalezen nebyl. Při importu bude vytvořen nový záznam.

- *Nespárovatelné importní záznamy* – záznamy z ext. databáze, pro které párování se záznamy vlastní databáze nebylo možné provést. Při importu bude vytvořen nový záznam. Pro informace o tom, co může být příčinou nespárovatelnosti importního záznamu, viz odkaz *Podrobnosti o nespárovatelných záznamech*.

[23] Podmínky párování záznamů zaručují pouze, že spárované záznamy disponují stejnými autory typu *Autor*. Záznamy se mohou lišit skladbou autorů typu *Editor* nebo údaji autora ve vztahu ke konkrétnímu záznamu (pořadí, mentální podíl atd.).

Samostatná skupina je nespárovatelným záznamům vyhrazena proto, že jejich import může potenciálně způsobit vznik duplicit.

Při párování záznamů se importní záznamy párují se všemi záznamy ve vlastní databázi. Omezená varianta párování pouze s těmi záznamy vlastní databáze, jež vyhovují stejným vyhledávacím parametrům, jaké byly použity pro vyhledání importních záznamů, by zvýšila riziko vzniku duplicit (importní záznam mající protějška ve vlastní databázi by se s ním nespároval, pokud by protějšek mezi záznamy s nimiž se páruje chyběl, a došlo by k vytvoření nového duplicitního záznamu).

Importní záznamy párujeme se záznamy vlastní databáze výhradně formou 1:1. Jakmile je nějaký importní záznam nebo záznam vlastní databáze spárován, párovací algoritmus si jej přestává všímat a další možná spárování už nezkouší.

Normalizace importních hodnot probíhá jako při importu z textového souboru (viz sekci 3.9), může však obnášet specifické rysy jako přemapování výčtových hodnot externí databáze na hodnoty Agendy věd. výsledků nebo převod reprezentace kalendářního data do standardního formátu aplikace.

Nově vytvořeným záznamům je za vlastníka přidělen aktuální uživatel. V případě spárovaných záznamů lze import rozdílů provést pouze tehdy, je-li vlastníkem protějška ve vlastní databázi aktuálně přihlášený uživatel (uživatel je oprávněn přepisovat pouze data, jejichž je vlastníkem). V případě neproveditelnosti importu z důvodu nedostatečného oprávnění se v levé části tabulky s výpisem záznamů zobrazí ikonka vlastníka.

Hromadný výpis záznamů Rozhraní hromadného výpisu a detailu záznamu (vysouvá se tlačítkem *Podrobnosti*) se pro nespárované a nespárovatelné záznamy shoduje s rozhraním importu záznamů z textového souboru (viz sekci 3.9). Totéž platí i o možnosti automatického vytvoření ve vlastní databázi dosud neexistujících autorů pomocí dialogu skrytého pod tlačítkem *Vytvořit autory v Agendě*.

Specifičtější je rozhraní pro import spárovaných záznamů, jejichž údaje se importují do nalezeného protějška záznamu, existujícího ve vlastní databázi. Zaškrtávací políčka pod titulkem *Importované atributy* slouží ke kompletnímu vyřazení některého údaje z importu spárovaných záznamů. Při nezaškrtnutém políčku je import údaje zakázán, při zaškrtnutém povolen (jeho provedení však ještě závisí na detailní konfiguraci v podrobnostech konkrétního záznamu). Tabulka hromadného výpisu spárovaných záznamů zobrazuje údaje importního záznamu, nikoli údaje protějška ve vlastní databázi. Výjimku tvoří údaje identifikátor a vlastník, kterými importní záznam vůbec nedisponuje. Fakt, že se nějaký importní údaj zobrazuje v tabulce hromadného výpisu spárovaných záznamů, neznamená, že bude v každém případě importován – v případě vyřazení údaje z importu konkrétního nebo všech spárovaných záznamů (viz zaškrtávací pole v detailu záznamu nebo pod titulkem *Importované atributy*) zůstane ve vlastní databázi zachována původní hodnota.

Celkový počet aktuálně vybraných záznamů k importu udává číslo uvnitř tlačítka *Importovat vybrané*.

Detail spárovaného záznamu Detailní zobrazení spárovaného záznamu pod tlačítkem *Podrobnosti* tvoří opět klasické tři záložky, ale výpis údajů je oproti nespárovaným a nespárovatelným importním záznamům rozšířen o sloupec *Hodnota v Agendě*, obsahující údaje spárovaného záznamu ve vlastní databázi. Přepsání původní hodnoty protějška ve vlastní databázi normalizovanou importní hodnotou lze pro každý údaj ovlivnit samostatným zaškrtávacím políčkem. (Zároveň však musí být import daného údaje povolen globálně – viz pole pod titulkem *Importované atributy*.) Políčko v záhlaví tabulky zajistí zaškrnutí polí pro všechny údaje. Není-li zaškrtávací políčko u některého údaje zobrazeno, import provést nelze z důvodu ochrany původní hodnoty před nežádoucím přepsáním. Výpis konkrétních příčin (pro daný údaj příčinu identifikuje ikonka ve sloupci *Normalizovaná importní hodnota*):

- normalizace importní hodnoty selhala (ikona vykřičník)
- normalizovaná importní hodnota je prázdná (pro hodnoty jiného než výčtového typu; ikona minus)
- normalizovaná importní hodnota má význam neurčené hodnoty (pro hodnoty výčtového typu; ikona otazník)
- normalizovaná importní hodnota je totožná s hodnotou protějška v Agendě (ikona „fajfka“)

Protože titulů záznamu je obecně více a dva spárované záznamy mohou mít odlišný počet titulů v různých jazycích (podmínkou pro spárování je pouze shodnost určitých údajů pro tituly v původním jazyce), musíme před importem údajů o titulech provést (analogicky k párování záznamů) párování titulů spárovaných záznamů. Výpis na záložce *Tituly záznamu* je na základě toho rozdělen do čtyř částí:

- *Spárované importní tituly* – tituly importního záznamu, k nimž byl nalezen protějšek mezi tituly záznamu ve vlastní databázi. Import údajů proběhne do tohoto již existujícího protějška. Ke spárovaní titulů postačí, mají-li shodný jazyk (viz odkaz *Podrobnosti o párování titulů*).
- *Nespárované importní tituly* – tituly importního záznamu, k nimž žádný protějšek mezi tituly záznamu ve vlastní databázi nalezen nebyl. Při importu bude pro záznam ve vlastní databázi vytvořen nový titul.
- *Nespárovatelné importní tituly* – tituly importního záznamu, pro které párování s tituly záznamu ve vlastní databázi nebylo možné provést. Při importu bude pro záznam ve vlastní databázi vytvořen nový titul. Pro informace o tom, co může být příčinou nespárovatelnosti titulu importního záznamu, viz odkaz *Podrobnosti o nespárovatelných titulech*.

- *Nespárované tituly ve vlastní databázi* – hromadný výpis titulů záznamu ve vlastní databázi, které nebyly spárovány se žádným z titulů importního záznamu. (Výpis nemá význam pro import.)

Záložka *Autoři záznamu* obsahuje pouze výpis autorů záznamu ve vlastní databázi, protože import údajů o autorech v případě spárovaných importních záznamů neprobíhá (viz výše).

Ve výchozím stavu je import konfigurován jako úplný – importují se všechny záznamy se všemi svými tituly (příp. i autory) a z importu spárovaných záznamů není vyřazen žádný údaj.

Záznamy pro export Záložka *Záznamy nalezené v Agendě* zobrazuje záznamy vlastní databáze vyhledané ekvivalentním způsobem jako importní záznamy z externí databáze. Ekvivalentní způsob vyhledání zahrnuje použití stejných vyhledávacích parametrů i metod hledání (podpora zástupných znaků, hledání hodnoty od začátku údaje namísto přesné hodnoty atp.).

Výpis záznamů je rozdělen do dvou částí:

- *Spárované záznamy v Agendě* – záznamy vlastní databáze spárované s nějakým importním záznamem. (Výpis zobrazuje pouze spárované záznamy náležící do skupiny záznamů vlastní databáze vyhledaných ekvivalentním způsobem jako záznamy z externí databáze. Importní záznamy jinak mohly být spárovány s jakýmikoli záznamy vlastní databáze, ne pouze se záznamy patřícími do zmiňované skupiny.)

- *Nespárované záznamy v Agendě* – záznamy nespárované se žádným importním záznamem.

Možnosti exportu záznamů a způsob jejich výpisu jsou shodné s výpisem záznamů na titulní stránce.

3.11 Přehled oprávnění

Tabulka 4 shrnuje oprávnění uživatelů aplikace vzhledem k databázovým entitám. Entitou „autor“ se myslí samostatný autor ve Správě autorů, nikoli autor přiřazený konkrétnímu záznamu. Entita „uživatel“ představuje uživatelský účet. Za jeho vlastníka považujeme právě toho uživatele, kterého reprezentuje.

	administrátor	**vlastník**	**ostatní**
záznam	ano / ano	ano / ano	ne / ne
autor	ano / ano	ano / ano*	ne / ne
uživatel	ano / ano*	ano / ne	ne / ne

Tabulka 4: Oprávnění k editaci / mazání různých typů dat

Nejednoznačné případy jsou v tabulce označeny hvězdičkou (*). Vlastník autora je oprávněn k jeho smazání pouze v případě, že autor doposud nebyl přiřazen žádnému záznamu. Administrátor může mazat pouze uživatele, kteří nejsou vlastníky žádných záznamů, ani autorů.

V tabulce 5 jsou vypsána oprávnění týkající se jednotlivých údajů databázových entit. Oprávnění platí pro údaje obecně, tj. bez ohledu na to, u kterých entit jsou evidovány. Údaj „identifikátor“ označuje jakýkoli atribut s interním identifikátorem[24] *i_public*, „vlastník“ atribut s int. identifikátorem *i_username* a „veřejnost“ atribut s int. identifikátorem *i_public*. Druhá z tabulek se týká variabilních údajů, tj. těch, které konfiguruje administrátor aplikace. (Administrátorem je určována i veřejnost variabilních údajů.) Importováním se myslí import z textového souboru nebo import prováděný v rámci porovnání databází.

pevný údaj	**identifikátor**	**vlastník**	**veřejnost**
zobrazování	ano / ano / ano	ne / ano / ano	ne / ano / ano
vyhledávání	ano / ano / ano	ne / ne / ano	ne / ano / ano
exportování	ano / ano / ano	ne / ano / ano	ano / ano / ano
importování	ne / ne / ne	ne / ne / ne	ne / ano / ano

variabilní údaj	**veřejný**	**neveřejný**
zobrazování	ano / ano / ano	ne / ano / ano
vyhledávání	ano / ano / ano	ne / ano / ano
exportování	ano / ano / ano	ne / ano / ano
importování	ne / ano / ano	ne / ano / ano

Tabulka 5: Oprávnění nepřihlášeného uživatele / přihlášeného uživatele / administrátora vzhledem k evidovaným údajům

4 Manuál pro administrátora

4.1 Systémové požadavky

K provozování aplikace postačuje sada volně dostupných technologií.

- *Databázový server* – **MariaDB 5.5** nebo vyšší, resp. **MySQL 5.5** nebo vyšší[25].
  - Při použití databázové PHP třídy `ODBCConn`[26] je nutné nainstalovat driver **MariaDB Connector/ODBC**, resp. **MySQL Connector/ODBC**.
- *Webový server* – doporučen **Apache**, ale ne nezbytně.

[24] Viz sekci 2.2.2.
[25] Aplikace byla testována pro MySQL 5.6 a MariaDB 10.1.
[26] Viz sekci 2.5.2.

  - Z důvodu větší ochrany přenášených hesel a dalších dat se doporučuje používat protokol **HTTPS** (srv. PHP konstantu `HTTPDigestAuth::FORCE_HTTPS`)[27].

- *PHP* – **PHP 5.5** nebo vyšší[28]. V rámci PHP jsou vyžadována tato rozšíření:
  - **MySQLi** – pouze při použití databázové PHP třídy `MySQLiConn`[29]; podporovány jsou drivery *libmysqlclient* i *mysqlnd*.
  - **ODBC** – pouze při použití databázové PHP třídy `ODBCConn`[30].
  - **cURL** s podporou protokolu HTTPS[31].
  - další rozšíření obvykle automaticky dostupná a nevyžadující konfiguraci (DOM, SimpleXML, PCRE, mbstring, Zip...)

Prostorová a paměťová náročnost aplikace závisí na představě provozovatele o množství evidovaných dat a frekventovanosti přístupů na server. Holá webová aplikace má zanedbatelnou velikost. Kritickými funkcemi co se hardwarové náročnosti jsou importy dat z textového souboru a porovnání s externími databázemi.

4.2 Instalace

Zprovoznění aplikace na serveru vyžaduje provedení následujících kroků:

1. Připravit databázi. (Jako výchozí kódování se důrazně doporučuje použít UTF-8.)
   (a) Vytvořit aplikační databázi SQL příkazem `CREATE DATABASE jmeno_databaze;`.
   (b) Vytvořit databázové tabulky provedením SQL příkazů v souboru `src/sql/create_table.sql`, přičemž v parametru příkazu `USE` bude uvedeno aktuální `jmeno_databaze`. (Vytvořené tabulky zatím obsahují pouze pevné atributy.)
   (c) Vložit do tabulky uživatelů uživatele se jménem „admin“[32], a to provedením SQL příkazu v souboru `src/sql/insert_users.sql`, přičemž o parametru příkazu `USE` platí totéž co v kroku 1b. (Uživateli admin tím bude přiděleno heslo „tuHeKe7“ a email „admin@localhost“.

[27] Z hlediska bezpečnosti přenášených hesel při nepoužití HTTPS rozlišujeme dva případy. V případě požadavku na autentizaci uživatele závisí ochrana na použité autentizační PHP třídě – viz sekci 2.5.2. V případě požadavků týkajících se správy uživatelských účtů (změny hesel, vytváření nových uživatelů) jsou hesla zasílána nechráněně (jako prostý text).

[28] Aplikace byla testována pro PHP 5.5 a PHP 5.6.

[29] Viz sekci 2.5.2.

[30] Viz sekci 2.5.2.

[31] Šifrovaný HTTPS přenos se využívá při získávání dat z externí databáze OBD UPOL (viz sekci 2.5.6), kdy je nutné před stažením exportu zadat přihlašovací údaje.

[32] Tj. uživatele s úrovní oprávnění *administrátor*.

Tyto hodnoty je doporučeno změnit ihned po prvním přihlášení volbou *Upravit účet*.)

(d) Vytvořit databázové uživatele a přidělit jim příslušná oprávnění provedením SQL příkazů v souboru `src/sql/create_grant_user.sql`, přičemž o parametru příkazu `USE` platí totéž co v kroku 1b[33]. Z bezpečnostních důvodů se doporučuje použít pro uživatele jiná přístupová hesla než jsou uvedena ve zmiňovaném souboru. Použití odlišných hesel si vyžádá pozdější úpravu těchto hesel v direktivách *password_** konfiguračního souboru `src/conf/application.ini.php`.

2. Připravit konfigurační soubory. Ve výchozím tvaru konfigurace umožňuje rozsáhlou evidenci publikačních vědeckých výsledků, včetně jejich exportů a porovnání s externími databázemi (více v příloze A). Obvyklý postup konfigurování:

   (a) Konfigurace aplikačního nastavení (viz sekci 4.3.1).

   (b) Konfigurace databázového schematu (viz sekci 4.3.2).

   (c) Úprava zbylé části konfigurace na základě změn v konfiguraci databázového schematu (viz ostatní sekce kapitoly 4.3).

3. Zkopírovat veškerý obsah adresáře `src` s výjimkou složky `src/sql` do příslušné složky na webovém serveru.

4. Otevřít webovou aplikaci v prohlížeči a přihlásit se s uživatelským jménem „admin“ a heslem uvedeným v kroku 1c.

5. Volbou z nabídky *Nástroje správce – Aktualizace databázového schematu* přejít na stránku umožňující vytvořit v databázi variabilní atributy definované v konfiguračních souborech. Vytvořit atributy pro všechny uvedené tabulky.

Pokud aplikace po provedení instalačních kroků nepracuje správně, může to být proto, že některé výchozí údaje v konfiguračních souborech nebo části kódu již nejsou aktuální. To se týká především návaznosti na externí databáze RIV (IS VaVaI) a OBD UPOL (funkcionalita porovnání s externími databázemi). Pro více informací viz sekci o údržbě (4.4).

4.3 Konfigurace

Následující sekce popisuje konfigurační soubory aplikace. Není-li uvedeno jinak, soubory používají syntaxi **INI** souboru. Z důvodu zabezpečení (ochrana před nepovolaným přístupem z Internetu) vyhovují **INI** soubory zároveň i syntaxi jazyka PHP (srv. začátek souboru s příkazem `die();`).

[33] V případě nedostupnosti většího množství databázových uživatelů lze použít jediného uživatele, viz sekci 2.2.3.

Kromě konfiguračních souborů je potřeba věnovat pozornost i některým konstantám v kódu aplikace (viz sekci 2.4) a konfiguračním direktivám PHP (např. direktivy *max_execution_time* a *memory_limit*, jejichž příliš nízká hodnota může znemožnit provedení náročnějších operací, nebo direktivy *upload_max_filesize* a *post_max_size* omezující maximální objem dat při importu z textového souboru).

Hlavičku a patičku zobrazovanou ve webovém rozhraní lze upravit v rámci PHP funkcí `HeaderFooter::generate_header()` (hlavička) a `HeaderFooter::generate_footer()` (patička).

Výchozí třídící kritéria pro tabulku s výpisem záznamů na titulní stránce se nastavují v PHP funkci `FilterSearch::data_sorting_default()`.

Způsob autentizace a připojení k databázi ovlivňují zvolená autentizační a databázová třída (viz sekci 2.5.2).

Je-li dokumentace v uživatelském rozhraní závislá na konfiguračních souborech, obnovuje se obvykle automaticky po jejich editaci. To platí pro informace o podmínkách párování při porovnání s externími databázemi nebo při importu z textového souboru (konf. soubory `src/conf/compare_*_pair.ini.php`, `src/conf/author_pair.ini.php`) i pro dokumentaci šablony bibliografického exportu pro vlastní LaTeX (konf. soubory `src/conf/table_*.ini.php`).

4.3.1 Aplikační nastavení (application.ini.php)

Popis direktiv v konfiguračním souboru `src/conf/application.ini.php`:

- *debug_mode* (bool) – zapíná režim ladění se zobrazováním PHP chyb (srv. PHP direktivu *display_errors*) a výpisem ladících informací při vzniku výjimky. Doporučuje se nepoužívat při ostrém nasazení aplikace, protože zobrazované informace odhalují uživateli interní uspořádání programu.
- *username_*, password_** (string) – přístupové údaje pro připojení k databázi. Tři databázoví uživatelé odpovídají třem úrovním oprávnění v aplikaci, čtvrtý uživatel slouží pro potřeby autentizace. Více v sekci 2.2.3.
- *database* (string) – jméno databáze s daty aplikace.
- *table_** (string) – databázové názvy tabulek s daty aplikace.
- *files_import_riv, files_import_obd* (string) – složky pro stažené exporty z databáze RIV (IS VaVaI) / OBD UPOL a související data. PHP musí mít pro uvedené složky práva čtení a zápisu. Cesta ke složce se uvádí vždy s lomítkem na konci. Nepoužívané soubory ve složkách jsou automaticky mazány, a proto se do těchto složek nesmí umístit žádná vlastní data.
- *textfile_templates* (string) – složka šablon pro import a export do textového souboru. Cesta ke složce se uvádí vždy s lomítkem na konci.
- *biblio_inis* (string) – složka konfiguračních souborů pro bibliografické exporty. Cesta ke složce se uvádí vždy s lomítkem na konci.

- *obd_username, obd_password* (string) – přihlašovací údaje do neveřejného rozhraní databáze OBD UPOL (rozhraní je dotazováno za účelem provedení exportu – funkcionalita porovnání s externí databází).
- *max_filesize_** (integer) – maximální velikost souborů s importními daty v MB. Týká se importu z textového souboru a porovnání s externími databázemi RIV (IS VaVaI) a OBD UPOL. V případě RIV (IS VaVaI) se omezení týká velikosti stahovaného komprimovaného souboru, nikoli vlastních dat (velikost dat po dekomprimaci je typicky násobně větší).
- *pagination_limit* (integer) – maximální počet zobrazovaných záznamů v tabulce s výpisem záznamů na titulní stránce.
- *order_columns_max* (integer) – maximální počet současně aplikovaných třídících kritérií pro tabulku s výpisem záznamů na titulní stránce.
- *imchar_gui_rows* (integer) – výška vstupního pole pro hodnotu atributu interního typu *imchar* (víceřádkový řetězec) v řádcích.
- *export_count_limit* (integer) – maximální počet současně exportovaných záznamů.
- *csv_separator* (string) – oddělovač sloupců v exportu do CSV.

4.3.2 Databázové schema (table_*.ini.php)

Mapování interních identifikátorů[34] na databázové názvy atributů se definuje v PHP třídách `Columns` a `ColumnsUsers`. Při změně názvu databázového atributu je nutné upravit i příslušné mapování.

Variabilní atributy databázového schematu lze konfigurovat v souborech:

- `src/conf/table_data.ini.php` (pro tabulku záznamů)
- `src/conf/table_title.ini.php` (pro tabulku titulů)
- `src/conf/table_author.ini.php` (pro tabulku autorů)
- `src/conf/table_data_author.ini.php` (pro vazební tabulku autorů)

Konfigurace každého ze souborů funguje na stejném principu. Schema tabulky uživatelů obsahuje pouze pevné atributy a proto konfigurovatelné není.

Konfigurační soubory aplikaci informují o existenci variabilních atributů a o způsobu, jak s nimi nakládat. Atribut nově definovaný v konf. souboru lze v databázi automatizovaně vytvořit (volba nabídky *Nástroje správce – Aktualizace databázového schematu*, viz sekci 4.6.1).

Každý konfigurační soubor je tvořen posloupností definic atributů. Jednotlivé definice začínají uvedením názvu atributu a dále obsahují proměnlivý počet parametrů. Parametry mohou obsahovat jednu hodnotu, nebo pole hodnot. Pořadí uvedení definic atributů ovlivní pořadí jejich výpisu v detailním zobrazení.

[34] Viz sekci 4.3.2.

```
1 [nazev_atributu]
2 parametr1 = hodnota_parametru1
3 parametr2 = hodnota_parametru2
4 parametr3[] = hodnota1_parametru3
5 parametr3[] = hodnota2_parametru3
6 ...
```

Zdrojový kód 4: Syntaxe definice atributu v konfiguračním souboru databázového schematu

Název atributu Názvy variabilních atributů (tj. všech, jež jsou definovány v konfiguračních souborech) začínají obvykle předponou *var_*. Název atributu uvedený v konfiguračním souboru je shodný s jeho databázovým názvem. Pro název atributu je zakázáno použít:

- název existujícího pevného atributu (obvykle začíná předponou *fix_*)
- název začínající dvojitým podtržítkem
- řetězec použitý pro danou tabulku jako interní identifikátor (obvykle začíná předponou *i_*; pro přehled int. identifikátorů viz PHP třídu `Columns`)

Interní typ	**Databázový typ**	**Popis**
iint	`INT`	Celé číslo v rozsahu od -2147483648 do 2147483647.
ismallint	`SMALLINT`	Celé číslo v rozsahu od -32768 do 32767.
iuint	`INT UNSIGNED`	Celé číslo v rozsahu od 0 do 4294967295.
iusmallint	`SMALLINT UNSIGNED`	Celé číslo v rozsahu od 0 do 65535.
ichar	`VARCHAR`	Řetězec povolené délky bez odřádkování.
imchar	`VARCHAR`	Řetězec povolené délky bez po sobě následujících odřádkování (řetězce s vícenásobným odřádkováním by narušily strukturu textového souboru pro import a export).
idate	`DATE`	Platné datum ve tvaru dd.mm.rrrr.
ienum	`ENUM`	Výčtový typ.

Tabulka 6: Interní datové typy atributů

Parametry atributu Skladba parametrů uváděných v definici atributu se liší podle toho, jaký je danému atributu v parametru *type* přiřazen interní datový typ (viz tab. 6). (Hvězdička (*) v seznamu parametrů značí parametry, jejichž uvedení je pro daný interní typ povinné.)

1. Parametry společné pro všechny interní datové typy (s výjimkou typu *ienum*, pro který se neuvádí parametry *placeholder* a *nullable*):

- *type** (string) – interní datový typ atributu (možné hodnoty uvedeny v tab. 6).
- *label** (string) – zobrazovaný název atributu.
- *description** (string) – popisek atributu (zobrazuje se po najetí myši na název atributu).
- *separator* (string) – pokud je parametr uveden, při výpisu atributů v detailním zobrazení bude v aktuální části výpisu zahájena nová skupina atributů, uvozená nadpisem obsahujícím hodnotu tohoto parametru.
- *placeholder* (string) (neuvádí se pro typ *ienum*) – nápověda pro vyplnění hodnoty atributu zobrazovaná na pozadí vstupního pole.
- *class* (string) (pouze pro tabulku záznamů) – specifikuje druhy záznamu, pro které se očekává (ale ne nutně) vyplnění atributu. Vstupní pole atributu se zvýrazní v případě, že aktuálně vyplňovaný záznam má jako svůj druh zvolený právě některý z druhů uvedených v parametru *class*.

  Hodnotou parametru se zadává jako výpis tříd oddělených mezerami. Třídy odpovídají druhům záznamů ve smyslu javascriptové funkce `enum_type_to_class()` v souboru `src/insert.php` (třídy jsou návratové hodnoty funkce). Příklad: chceme-li pole s atributem zvýraznit pro typy záznamu článek v časopisu a příspěvek ve sborníku, v parametru *class* uvádíme hodnotu „j d“.
- *public** (bool) – veřejnost atributu (určuje, má-li být zobrazen i nepřihlášenému uživateli).
- *extended** (bool) – určuje, má-li být atribut přítomen i v základním vyhledávání (jinak pouze v rozšířeném).
- *nullable** (bool) (neuvádí se pro typ *ienum*) – určuje, může-li být hodnota atributu prázdná.

  Z důvodu efektivního průběhu importů z textového souboru a porovnání s externími databázemi se nedoporučuje deklarovat jako nulovatelné ty atributy, jejichž hodnoty v importních bývají obvykle prázdné anebo vůbec neexistují[35]. V obou případech se za importní hodnotu vezme prázdná hodnota, jejíž normalizace v případě, že atribut není nulovatelný, selže.
- *sortable** (bool) – tříditelnost atributu[36].

[35] Tzn. v případě importu z textového souboru neexistuje příslušná proměnná v šabloně textového souboru a v případě porovnání s externí databází neexistuje mapování atributu na data exportu externí databáze.

[36] Týká se uživatelsky zadaného třídění v tabulce se záznamy na titulní stránce, nikoli třídění kdekoli v aplikaci.

- *default* (string) – výchozí hodnota atributu. Bude použita pro import v případě, že normalizace původní importní hodnoty selhala. Neuvedení parametru signalizuje prázdnou výchozí hodnotu. Prázdná hodnota se vždy uvádí tímto způsobem, nikdy (ani pro řetězcové typy) uvedením parametru s hodnotou prázdného řetězce (ten lze použít jedině u interního typu *ienum*, je-li prázdný řetězec jeho výčtovou hodnotou)[37]. Uvedená výchozí hodnota musí být pro daný atribut zcela validní, tj. musí respektovat interní datový typ a na něj navazující parametry a omezení nulovatelnosti (parametru *nullable*). Atribut interního typu *ienum* není nulovatelný nikdy, a proto musí mít vždy definovaný parametr *default* (byť s hodnotou prázdného řetězce).

  Pro atribut interního typu *ienum* je nejtypičtější výchozí hodnotou právě prázdný řetězec, mající význam neurčené hodnoty. Více informací v poznámce u parametru *values[]*.
- *default_new* (string) – předvyplněná hodnota při interaktivním vkládání nové položky. Pro vyplnění parametru platí stejná pravidla jako pro *default*, s tím ale, že jej lze v jakémkoli případě vynechat.

2. Typově specifické parametry:

   - *length** (integer) (typy *ichar*, *imchar*) – maximální délka řetězce.
   - *values[]** (pole stringů) (typ *ienum*) – seznam povolených výčtových hodnot.

     Poznámka k neurčené výčtové hodnotě: Z důvodu efektivního průběhu importů z textového souboru a porovnání s externími databázemi se často vyplatí zařadit do výčtu i hodnotu prázdného řetězce, mající význam neurčené hodnoty.

     – Neurčená hodnota se bere za importní v případě, že hodnota daného atributu v importních datech je prázdná anebo vůbec neexistuje[38]. Uvedením neurčené hodnoty ve výčtu předejdeme selhání normalizace takové importní hodnoty.

     – Pokud při porovnání databází importujeme rozdíly do již existujících dat ve vlastní databázi, neurčená hodnota představující pro daný výčtový atribut normalizovanou importní hodnotu jako jediná původní hodnotu atributu ve vlastní databázi nepřepíše. Označíme-li tedy neurčenou hodnotu v parametru *default* za výchozí hodnotu daného atributu, v případě selhání normalizace importní hodnoty se normalizovanou importní hodnotou stane výchozí neurčená hodnota a k nežádoucímu přepisu dat nedojde.

[37] To odpovídá způsobu reprezentace řetězcových a výčtových hodnot v databázi – viz sekci 2.2.4.

[38] Viz poznámku č. 35.

Pokud pro pro daný atribut hodnota v importních datech neexistuje nebo je prázdná (srv. předchozí odrážku), za importní hodnotu se vždy bere neurčená hodnota. Je-li tato platnou výčtovou hodnotou, její normalizace jakožto importní hodnoty neselže a podobně jako v předchozím případě nedojde ani k nežádoucímu přepisu dat.

Na výčtové hodnoty některých atributů tabulky *„data_author“* odkazují globální PHP konstanty `AUTHOR_*`. Hodnota uvedená v těchto konstantách musí být platnou výčtovou hodnotou daného atributu (více v komentářích zdroj. kódu `src/load/constants.php`).

- *order_alpha** (bool) (typ *ienum*) – určuje, má-li třídění atributu probíhat podle abecedy, anebo podle pořadí uvedení výčtových hodnot v konfiguračním souboru.

Úpravy v databázi již existujících atributů Skript na stránce *Nástroje správce – Aktualizace databázového schematu* umožňuje na základě konfiguračních souborů vytvořit v databázi dosud neexistující atribut. Změny konfigurace již existujících atributů však nijak nereflektuje. Pokud byl tedy v konfiguraci atributu změněn některý jeho parametr, který souvisí s databázovou definicí atributu (jedná se o parametry *type*, *nullable*, *length* a *values[]*), nebo i samotný název atributu, je nutné příslušné úpravy atributu v databázi provést ručním SQL příkazem. Stejně i v případě smazání definice atributu v konfiguračním souboru je nutné odstranit atribut z databázového schematu ručně.

Výjimku představuje přidání (nikoli odebrání nebo modifikace) výčtových hodnot atributu interního typu *ienum* v parametru *values[]*. V tomto případě lze opět použít vestavěnou funkcionalitu *Aktualizace databázového schematu*.

Hodnoty atributů interního typu *ienum* se při nastaveném příznaku v parametru *order_alpha* třídí podle pořadí uvedení výčtových hodnot v parametru *values[]*. Rozhodující je však pořadí uvedených hodnot v okamžiku vytváření atributu v databázi. Dodatečné změny pořadí nebudou mít na způsob třídění vliv.

Atributy se zvláštním významem Zvláštní význam má v aplikaci každý variabilní atribut definovaný v konfiguračním souboru, na který odkazuje nějaký interní identifikátor. Takový atribut je v konf. souboru ve výchozím stavu uvozen komentářem se jménem odpovídajícího interního identifikátoru. Pro přehled všech variabilních atributů se zvláštním významem viz PHP třídu `Columns` a všechny tam uvedené atributy vyjma pevných.

V případě změny názvu atributu se zvláštním významem je nutné aktualizovat mapování příslušného interního identifikátoru v PHP třídě `Columns` tak, aby se mapoval na aktuálně platný název atributu. Bez korektního mapování interních identifikátorů na názvy atributů nebude aplikace fungovat správně.

Měnit konfiguraci atributů se zvláštním významem často není doporučeno. Pro více informací viz komentáře v kódu třídy `Columns` a komentáře u definic atributů v konf. souboru příslušné tabulky.

Výchozí hodnotu (parametry *default* a *default_new*) atributu *i_year („data“)* (rok uplatnění) je nutné vždy nastavit ručně (s příchodem nového kalendářního roku nedochází k automatické aktualizaci).

4.3.3 Zobrazení (interface.ini.php, interface_ienums.php)

Zobrazované variabilní atributy Konfigurační soubor `src/conf/interface.ini.php` vybírá a určuje pořadí variabilních atributů zobrazených v tabulkách hromadného výpisu položek (např. v tabulce se záznamy na titulní stránce, tabulce s výpisem autorů záznamu v detailu a při vkládání záznamu atd.). Při detailním výpisu (např. v detailu záznamu nebo autora, při jejich vkládání, nebo v rozhraní pro rozšířené vyhledávání) se zobrazují vždy všechny atributy, a to v pořadí, v jakém byly uvedeny v konf. souboru `src/conf/table_*_.ini.php` příslušné tabulky.

Soubor `src/conf/interface.ini.php` obsahuje tyto direktivy (v kulaté závorce uveden kód databázové tabulky, jejíž atributy se uvádí, v hranaté umístění zobrazované tabulky, na niž má direktiva vliv):

- *results_data[]* (*„data“*) – atributy zorazené v tabulce s výpisem záznamů na titulní stránce [index.php]
- *results_title[]* (*„title“*) – atributy zobrazené v tabulce s výpisem titulů záznamu [insert.php, detail.php]
- *results_data_author_x[]* (*„author“*),
  results_data_author[] (*„data_author“*) – atributy zobrazené v tabulce s výpisem autorů záznamu [insert.php, detail.php]
- *results_author[]* (*„author“*) – tabulka s výpisem všech autorů Správy autorů [author_manage.php]
- *import_results_data[]* (*„data“*) – atributy zobrazené v tabulce s výpisem záznamů pro import nebo export [import_text.php, import_compare.php]
- *import_results_title[]* (*„title“*) – atributy zobrazené v tabulce s výpisem titulů importního záznamu [import_ajax.php]
- *import_results_data_author_x[]* (*„author“*),
  import_results_data_author[] (*„data_author“*) – atributy zobrazené v tabulce s výpisem autorů importního záznamu [import_ajax.php]

U každé direktivy se povinně uvádí alespoň jeden atribut. Některé atributy se implicitně zobrazují specifickým způsobem (např. atribut *i_riv („data“)* se zobrazí jako odkaz do webového rozhraní systému RIV). Pro detaily viz PHP funkci `ResultsPage::go_col_REST()`.

Zobrazování výčtových hodnot V uživatelském rozhraní obvykle nechceme zobrazovat výčtové hodnoty v podobě uvedené v parametru *values[]* v konf. souboru `src/conf/table_*.ini.php`. Hodnotu výčtu pro zobrazení v uživatelském rozhraní lze definovat v souboru `src/conf/interface_ienums.php`. Jde o soubor v syntaxi jazyka PHP, obsahující jedinou proměnnou `$INTERFACE_IENUMS`. V této proměnné je uloženo vícerozměrné asociativní pole se strukturou odpovídající čtyřem databázovým tabulkám. Samotné mapování výčtových hodnot na řetězce zobrazované uživateli je definováno v nejhlubší úrovni vnoření.

```
$INTERFACE_IENUMS = array(
   "data" => array(
     nazev_vyctoveho_atributu1 => array(
       vyctova_hodnota1 => zobrazovany_retezec1,
       vyctova_hodnota2 => zobrazovany_retezec2,
     ...
     ),
     ...
   ),
   "title" => array(
     ...
   ),
   "author" => array(
     ...
   ),
   "data_author" => array(
     ...
   )
);
```

Zdrojový kód 5: Struktura souboru `src/conf/interface_ienums.php`

V případě, že mapování pro danou výčtovou hodnotu anebo pro celý atribut výčtového typu není definováno, dojde ke zobrazení původní výčtové hodnoty. Pro výčtové hodnoty prázdného řetězce je doporučeno vždy definovat neprázdný zobrazovaný řetězec, a to za účelem zachování validity HTML[39]. Vzhledem k tomu, že výčtová hodnota prázdného řetězce má význam neurčené hodnoty (viz sekci 2.2.4), obvykle se pro ni definuje zobrazovaný řetězec „(neurčeno)“.

Abecední třídění atributů výčtového typu probíhá vždy podle původních výčtových hodnot, nikoli podle zobrazovaných řetězců. Pro zachování správnosti třídění vzhledem ke zobrazovaným řetězcům se proto doporučuje volit řetězce ve tvaru obsahujícím na začátku původní výčtovou hodnotu (např. „vyctova_hodnota1 - popis_vyctove_hodnoty1“).

[39]Validní HTML nepodporuje v elementu `<option>` zobrazování prázdné hodnoty.

4.3.4 Porovnání s externími databázemi (compare_*.ini.php)

Konfigurace porovnání s externími databázemi RIV (IS VaVaI) a OBD UPOL závisí na proměnlivém tvaru exportů těchto databází. Proto je nutné věnovat konfiguraci pozornost nějen při instalaci, ale i během provozování aplikace (viz sekci 4.4).

Pro porovnání s databází RIV (IS VaVaI) využíváme její **DBF** export (srv. sekci 1.4.1), pro OBD UPOL **XML** export (srv. sekci 1.4.2).

4.3.4.1 Mapování atributů (compare_*_cols.ini.php)

Atributy jednotlivých tabulek mapujeme na data exportu z externí databáze v souborech `src/conf/compare_*_cols.ini.php`. Struktura souborů odpovídá čtyřem tabulkám aplikační databáze (viz zdroj. kód 6). Význam uváděných identifikátorů dat se liší podle typu exportu, na který mapujeme.

```
1  [data]
2  nazev_atributu1 = identifikator_dat1
3  nazev_atributu2 = identifikator_dat2
4  ...
5  [title]
6  ...
7  [author]
8  ...
9  [data_author]
10 ...
```

Zdrojový kód 6: Struktura souboru `src/conf/compare_*_cols.ini.php`

Mapování lze definovat (až na výjimky) pro jakékoli variabilní atributy a pro pevný atribut *i_public („data“)*. Není-li mapování pro některý mapovatelný atribut definováno, jako importní hodnota se pro něj bere neurčená hodnota (prázdný řetězec) pro interní typ *ienum* a prázdná hodnota pro všechny ostatní typy. Atributy, jejichž mapování má během procesu porovnávání s externí databází zvláště důležitý význam, jsou vypsány v komentáři v konfiguračním souboru.

Mapování pro databázi OBD UPOL V případě souboru `src/conf/compare_obd_cols.ini.php` (mapování na data **XML** exportu databáze OBD) odpovídají identifikátory dat názvům XML elementů. Pro atributy tabulky *„data“* uvádíme názvy XML elementů, jež jsou přímými potomky elementu `<zaznam>` a samy žádné další potomky neobsahují. Pro atributy tabulky *„title“* platí totéž s tím, že uvádíme přímé potomky elementu `<titul>`. Pro atributy tabulek *„author“* a *„data_author“* platí opět totéž, v obou případech však uvádíme přímé potomky elementu `<autor>`.

Mapování nelze definovat pro pevný atribut *i_public („data“)* a pro variabilní atributy *i_language_orig („data“)* a *i_url („data“)* – tyto atributy se mapují

nestandardním způsobem (viz PHP funkci `DataGetOBD::get()` a konstanty třídy `DataGetOBD`).

Mapování pro databázi RIV (IS VaVaI) Struktura konf. souboru `src/conf/compare_riv_cols.ini.php` (mapování na data **DBF** exportu databáze RIV) se poněkud liší od struktury zachycené ve zdroj. kódu 6. Identifikátory dat odpovídají sloupcům v **DBF** exportu. Vzhledem k tomu, že hodnotě jednoho atributu obvykle odpovídá nikoli hodnota jednoho sloupce, nýbrž zřetězení hodnot několika sloupců, tak také každému atributu odpovídá ne jeden identifikátor dat, nýbrž pole identifikátorů. Odpovídá-li tedy hodnota atributu zřetězení hodnot dvou **DBF** sloupců, píšeme v konfiguraci `nazev_atributu1[] = identifikator1_dat1` a `nazev_atributu1[] = identifikator2_dat1`, kde identifikátory dat jsou názvy příslušných **DBF** sloupců. Hodnoty se zřetězí v pořadí, v jakém byly uvedeny.

V konfiguračním souboru lze definovat pouze mapování atributů tabulky *„data“*. Atributy ostatních tabulek nelze přímočaře mapovat na **DBF** sloupce, protože aplikace provádí netriviální abstrakci nad **DBF** tabulkou. Názvy **DBF** sloupců, jejichž hodnoty tato abstrakce používá, jsou určeny konstantami PHP třídy `DataGetRIV`. Z atributů tabulky *„data“* nelze dále mapovat pevný atribut *i_public („data“)*. Pro podrobnosti o nestandardním mapování atributů viz PHP funkci `DataGetRIV::get()`.

4.3.4.2 Mapování hodnot výčtů (compare_*_ienums.php)

Výčtové hodnoty v datech exportu z externí databáze se často liší od výčtových hodnot Agendy vědeckých výsledků. V případě importu nepovolených hodnot výčtového typu dochází k selhání jejich normalizace. Proto je nutné definovat mapování výčtových hodnot externí databáze na výčtové hodnoty Agendy vědeckých výsledků, k čemuž slouží soubor `src/conf/compare_*_ienums.php`. Jde o soubor v PHP syntaxi, obsahující jedinou proměnnou `$COMPARE_RIV_IENUMS`, resp. `$COMPARE_OBD_IENUMS`. Obsahem proměnné je vícerozměrné asociativní pole s totožnou strukturou jako v případě souboru `src/conf/interface_ienums.php` (viz zdroj. kód 5). Definice mapování jsou tedy ve tvaru `externi_vyctova_hodnota => vyctova_hodnota_Agendy`.

V případě, že mapování pro danou výčtovou hodnotu anebo pro celý atribut výčtového typu není definováno, pro import se použije původní výčtová hodnota. Kromě mapování 1:1 lze použít i mapování M:1 (více výčtových hodnot externí databáze mapujeme na totožnou výčtovou hodnotu vlastní databáze).

4.3.4.3 Párování záznamů (compare_*_pair.ini.php)

Konfigurační soubor `src/conf/compare_*_pair.ini.php` určuje, na jakých atributech (tzv. *párovací atributy*) se musí importní záznam a záznam z vlastní databáze shodovat, aby mohly být považovány za totožné (tzv. *spárované záznamy*). Strukturu konf. souboru znázorňuje zdroj. kód 7.

```
1  title[] = nazev_atributu1
2  title[] = nazev_atributu2
3  ...
4
5  author[] = nazev_atributu3
6  ...
7
8  data_author[] = nazev_atributu4
9  ...
10
11 [data]
12 druh_zaznamu1[] = nazev_atributu5
13 druh_zaznamu1[] = nazev_atributu6
14 ...
15 druh_zaznamu2[] = ...
16 ...
```

Zdrojový kód 7: Struktura souboru `src/conf/compare_*_pair.ini.php`

Direktiva `title[]` obsahuje atributy tabulky titulů, které musí být shodné pro tituly spárovaných záznamů v původním jazyce (tj. pro ty tituly spárovaných záznamů, jejichž jazyk je shodný s původním jazykem záznamu, k němuž náleží). Direktiva `author[]`, resp. `data_author[]` určuje atributy (vazební) tabulky autorů, jež musí být shodné, aby autor importního záznamu mohl být považován za totožného s autorem záznamu ve vlastní databázi (pro spárování záznamů je nutné, aby každý autor typu *Autor* v importním záznamu i v záznamu z vlastní databáze měl ve spárovaném záznamu totožného protějška).

Sekce `[data]` obsahuje sadu direktiv uvozených výčtovou hodonotou druhu záznamu (atribut *i_type („data“)*). V každé direktivě jsou uloženy atributy tabulky záznamů, jež musí být pro spárované záznamy shodné v případě, že druh importního spárovaného záznamu se rovná názvu direktivy. Uvedením `D[] = "var_rok"` tedy např. zajistíme, že importní záznam druhu D bude spárován pouze s těmi záznamy, jež mají stejnou hodnotu roku uplatnění. Druh párovaného záznamu ve vlastní databázi nehraje při výběru párovacích atributů žádnou roli.

Do direktiv zapisujeme názvy atributů Agendy vědeckých výsledků. Jejich mapování na data exportů externích databází definuje až soubor `src/conf/compare_*_cols.ini.php`. Uvedení direktivy není povinné pro žádnou z direktiv. Párování lze definovat pro jakékoli variabilní atributy a pro pevný atribut *i_public („data“)*. Přirozeně nemá smysl definovat párování pro atributy bez zavedeného mapování na data exportu z externí databáze.

Jména direktiv sekce `[data]` sice identifikují druhy importních záznamů, používáme pro ně však výčtové hodnoty Agendy vědeckých výsledků. Jsou-li totiž pro atribut druhu záznamu výčtové hodnoty externí databáze odlišné od hodnot Agendy věd. výsledků, typicky dochází k jejich přemapování na hodnoty Agendy věd. výsledků (viz sekci 4.3.4.2).

4.3.5 Párování autorů (author_pair.ini.php)

Při importu záznamů z textového souboru a při importu nespárovaných nebo nespárovatelných záznamů z externí databáze (funkce porovnání s externí databází) jsou autoři těchto záznamů porovnáváni s autory ve Správě autorů (tj. ve vlastní databázi)[40]. Pokud je ve vlastní databázi nalezen totožný protějšek autora, lze jej použít při importu bez nutnosti vytvářet ve Správě autorů autora nového. Konfigurační soubor `src/conf/author_pair.ini.php` určuje atributy tabulky autorů (nikoli vazební tabulky autorů!), na kterých se importní autor a autor ve vlastní databázi musejí shodovat, aby mohli být považováni za totožné. Atributy se definují v direktivě `author_pair[]`, která je jedinou direktivou tohoto konf. souboru.

```
1 author_pair[] = nazev_atributu1
2 author_pair[] = nazev_atributu2
3 ...
```

Zdrojový kód 8: Struktura souboru `src/conf/author_pair.ini.php`

Párování lze definovat pro jakékoli variabilní atributy. Přirozeně nemá smysl definovat párování pro ty atributy, kterým v šabloně importního textového souboru nepřísluší žádná proměnná, resp. (v případě porovnání s externí databází) nemají zavedeno mapování na data ext. databáze.

4.3.6 Bibliografické exporty

Tvar exportů ve formátech seznamů literatury určují konfigurační soubory ve složce `src/conf/biblio`. Výjimku tvoří export pro vlastní LaTeX, který konfigurační soubor nemá (lze však nastavit jeho výchozí šablonu, viz sekci 4.3.7, a názvy proměnných používaných v šabloně, viz sekci 4.3.8).

Export	**Konf. soubor**
BibTeX	`src/conf/biblio/bibtex.ini.php`
BibTeX - styl ČSN ISO 690	`src/conf/biblio/bibtex_csniso690.ini.php`
DBLP XML	`src/conf/biblio/dblp.ini.php`

Tabulka 7: Konfigurační soubory bibliografických exportů

Všechny bibliografické konfigurační soubory opisují tutéž strukturu, vycházející z principů systému BibTeX[41]. Jednotlivým *BibTeX typům* tedy přiřazují povinná a volitelná *BibTeX pole*. To by samo o sobě nemělo význam, a proto

[40] Toto porovnání nelze zaměňovat s porovnáváním atributů autorů při párování záznamů (viz sekci 4.3.4.3).

[41] Export pro citační styl ČSN ISO 690 představuje pouze přizpůsobenou variantu obvyklého BibTeX exportu a formát DBLP XML se liší z pohledu syntaxe, ne však z pohledu struktury.

dále definují mapování druhů záznamů na *BibTEX typy* a mapování *BibTEX polí* na databázové atributy. Pro více informací o bibliografickém formátu systému BibTEX viz sekci 1.5.2. Stručný popis formátu DBLP XML obsahuje sekce 1.5.3. Vzor bibliografického konf. souboru obsahuje zdroj. kód 9.

```
1  [_types]
2  druh_zaznamu1 = bibtex_typ1
3  druh_zaznamu2 = bibtex_typ2
4  ...
5
6  [_fields]
7  bibtex_pole1 = nazev_atributu1
8  bibtex_pole2 = nazev_atributu2
9
10 [bibtex_typ1]
11 required[] = bibtex_pole1
12 required[] = bibtex_pole2
13 ...
14 optional[] = bibtex_pole3
15 optional[] = bibtex_pole4
16 ...
17
18 [bibtex_typ2]
19 required[] = ...
20 ...
21
22 ...
```

Zdrojový kód 9: Struktura konfiguračního souboru pro bibliografický export

Sekce `[_types]` konf. souboru obsahuje mapování druhu záznamu na *BibTEX typ*, a to pro každý druh záznamu (tj. pro všechny výčtové hodnoty atributu *i_type („data“)*). Mapování M:1 je přípustné.

V sekci `[_fields]` je definováno mapování *BibTEX polí* na atributy tabulky záznamů. Uvádět lze výhradně variabilní atributy. Mapování nelze definovat pro *BibTEX pole* „author“, „editor“ a „title“, která se pevně daným způsobem mapují na atributy tabulky autorů nebo titulů (viz PHP funkci `ExportBiblio::export_field()`). Mapování M:1 je přípustné.

Následují sekce se jmény *BibTEX typů*. Svoji sekci musí mít minimálně všechny ty *BibTEX typy*, na něž se v sekci `[_types]` mapuje nějaký druh záznamu. Direktiva `required[]` určuje povinná, direktiva `optional[]` volitelná *BibTEX pole* daného *BibTEX typu*. Pole budou exportována v pořadí uvedení v konf. souboru.

Při vlastním exportu záznamu se nejprve v sekci `[_types]` najde *BibTEX typ* odpovídající jeho druhu. Na obsahu sekce se jménem nalezeného *BibTEX typu* pak závisí skladba exportovaných dat. *BibTEX pole* v direktivě `required[]`

budou exportována vždy, zatímco pole v direktivě `optional[]` pouze v případě neprázdnosti exportované hodnoty[42].

Bibliografické exporty lze provádět i v nepřihlášeném stavu – *BibTEX pole* namapovaná v sekci `[_fields]` na neveřejné atributy budou v takovém případě vyřazena z exportu. Vyřazení lze však z povahy věci provést jen pro pole direktivy `optional[]`. V direktivě `required[]` není proto uvádění polí namapovaných na neveřejné atributy povoleno.

Stejným způsobem jako v případě neveřejnosti se z exportu vyřazují i *BibTEX pole* bez definovaného mapování na atribut v databázi.

Výchozí konfigurace bibliografických exportů Výchozí podoba konfiguračních souborů se drží původní specifikace BibTEXu [4]. Konfigurace je totožná pro všechny tři druhy exportů (tvar exportu pro citační styl ČSN ISO 690 lze odstínit podle potřeby).

V případě potřeby lze v konfiguračních souborech dodefinovat další *BibTEX typy* a *BibTEX pole*, např. podle rozšířené specifikace balíku BibLATEX [5].

4.3.7 Šablony textového souboru

Šablonou textového souboru se rozumí opět textový soubor, určující pomocí speciální syntaxe strukturu textového souboru pro import nebo export dat. Šablony používají kódování UTF-8.

4.3.7.1 Správa šablon Pro šablony textového souboru je určena složka `src/conf/textfile_templates`. Ukládají se v ní tři typy šablon:

- šablony s příponou „.tmpl-public.php“[43] – veřejné šablony
- šablony s příponou „.tmpl-nonpublic.php“[44] – neveřejné šablony (dostupné pouze v přihlášeném stavu)
- šablona s názvem „tmpl-latex-default.php“[45] – výchozí šablona bibliografického exportu pro vlastní LATEX

Všechny šablony (s výjimkou šablony pro vlastní LATEX) mohou být použity pro import i export záznamů. Veřejné šablony by neměly odkazovat na žádné neveřejné údaje, jinak při exportu v nepřihlášeném stavu dojde k chybě.

Počet definovaných veřejných a neveřejných šablon může být libovolný (i nulový). Šablona pro vlastní LATEX musí být definována v každém případě.

4.3.7.2 Syntaxe šablony Syntaxi šablony názorně popíšeme na příkladu ve zdroj. kódu 10 (řetězec „EOF“ v závěru kódu symbolizuje konec souboru za posledním odřádkováním a do vlastní šablony nepatří).

[42] Výčtová hodnota prázdného rětězce není považována za prázdnou hodnotu.
[43] Přípona je určena PHP konstantou `TEMPLATE_PUBLIC_SUFFIX`.
[44] Přípona je určena PHP konstantou `TEMPLATE_NONPUBLIC_SUFFIX`.
[45] Název je určen PHP konstantou `TEMPLATE_LATEX_DEFAULT`.

```
1  <?php die(); ?>
2  separator = "========================"
3  PEVNÝ TEXT
4
5  Pevný text A: @promenna1
6  Pevný text B: @promenna2
7
8  Pevný text C: @promenna3 @promenna4
9
10 Pevný text D:#toto je komentář
11 @promenna5 @promenna6
12 %
13
14 Pevný text E:
15 @promenna7 @promenna8
16 %!
17
18 Pevný text F: @!promenna9
19 Pevný text G: [@!promenna10] [@!promenna11]
20 EOF
```

Zdrojový kód 10: Příklad šablony textového souboru

Řádek č. 1 v příkladu šablony obsahuje předem daný řetězec, zajišťující ochranu šablony před nepovolaným přístupem z Internetu.

Na řádku č. 2 je v uvozovkách definován tzv. *oddělovač*, tj. neprázdný řetězec oddělující v textovém souboru jednotlivé záznamy. Jako oddělovač by měl být zvolen řetězec dostatečně specifický na to, aby se nevyskytoval v textech šablony, ani v importovaných datech.[46]

Počínaje řádkem č. 3 až do konce souboru definuje šablona strukturu textového souboru pro import a export. Šablona musí být zakončena vždy alespoň jedním odřádkováním.

Ve vlastní šabloně můžeme rozlišit tři druhy textů:

- *Komentáře* – texty uvozené znakem „#“ a zakončené odřádkováním nebo koncem souboru. Před zpracováním šablony dojde k jejich odstranění, a proto nemají pro strukturu textového souboru žádný význam. Budou však zobrazeny uživateli v generovaném vzoru importního souboru (viz sekci 3.9.1).

- *Proměnné* – texty uvozené znakem „@“ nebo sekvencí znaků „@!“, přičemž za uvozením musí následovat neprázdná sekvence alfanumerických znaků bez diakritiky včetně podtržítek, nesoucí název proměnné. Proměnná končí tam, kde začíná první znak, který pro název proměnné není povolen.

[46] Porušení této podmínky může (za určitých okolností) vést k nesprávnému určení konce záznamu.

Proměnné uvozené sekvencí „@!“ označujeme jako nulovatelné proměnné, protože akceptují prázdné hodnoty.

- *Řetězce signalizující opakování vzoru* – takto jsou interpretovány znak „%“ a sekvence „%!“ za předpokladu, že došlo k naplnění následujících podmínek:
  - znak (sekvence) se nachází na samostatném řádku
  - předcházející řádek je neprázdný
  - za znakem (sekvencí) následuje dvojité odřádkování

  Sekvence „%!“ označuje nulovatelné opakování vzoru (vzor nemusí být použit ani jednou). Vzorem se myslí předcházející řádek šablony.
- *Pevné texty* – všechny ostatní texty.

V našem příkladu jsou jako řetězce signalizující opakování vzoru interpretovány znak „%“ na řádku č. 12 i sekvence „%!“ na řádku č. 16 (splňují všechny výše popsané podmínky). Za pevné texty se kromě řetězců „Pevný text...“ považují i všechny bílé znaky a také hranaté závorky na řádku č. 19.

Princip vytvoření importního nebo exportního textového souboru na základě šablony spočívá v nahrazení proměnných importními nebo exportními hodnotami, přičemž všechny pevné texty zůstávají beze změn. Takto vzniká jeden importní nebo exportní záznam. Po přidání oddělovače na závěr záznamu lze proces zopakovat a za oddělovačem začít další záznam. Analogicky pro všechny další záznamy.

Řetězce „%“ a „%!“, signalizující opakování vzoru (tj. opakování předchozího řádku šablony), umožňují opakovat v textovém souboru řádek v syntaxi předcházejícího řádku šablony, a to až do uvedení dvojitého odřádkování. V případě řetězce „%!“ nemusí být řádek v syntaxi vzoru uveden vůbec. Smyslem funkcionality opakování vzoru je umožnit import a export vícenásobných hodnot.

Příklad importního nebo exportního textového souboru, validního vzhledem k šabloně ve zdroj. kódu 10, obsahuje zdroj. kód 11.

Na řádcích č. 10 až 12 textového souboru byl dvakrát zopakován (tj. použit celkem třikrát po sobě) řádek v syntaxi řádku šablony č. 11 (opakování vzoru povolené řetězcem „%“ na řádku šablony č. 12). Řádek v syntaxi řádku šablony č. 15 nebyl naopak v textovém souboru použit ani jednou (řádek č. 15 textového souboru je prázdný) – toto nulové opakování vzoru umožnil řetězec „%!“ na řádku šablony č. 16.

Pro nulovatelnou proměnnou „promenna11“ byla na řádku č. 17 textového souboru použita prázdná hodnota. Povinné použití ohraničujících hranatých závorek v této situaci značně zpřehlednilo syntaxi.

Z hlediska importu je podstatné, že za hodnotu proměnné se v textovém souboru bere vše od pozice odpovídající pozici proměnné v šabloně až po první výskyt znaku, který v šabloně následuje těsně za proměnnou. Tímto znakem je

```
1  PEVNÝ TEXT
2
3  Pevný text A: viceslovna hodnota1
4  Pevný text B: radek1 hodnoty2
5  radek2 hodnoty2
6
7  Pevný text C: hodnota3 hodnota4
8
9  Pevný text D:
10 hodnota5-a hodnota6-a
11 hodnota5-b hodnota6-b
12 hodnota5-c hodnota6-c
13
14 Pevný text E:
15
16 Pevný text F: viceslovna hodnota9
17 Pevný text G: [viceslovna hodnota10] []
18 ========================
19 PEVNÝ TEXT
20 (...)
```

Zdrojový kód 11: Příklad textového souboru se strukturou podle šablony ve zdroj. kódu 10

typicky jednoduché odřádkování (hodnota proměnné může být víceslovná) nebo mezera (výhradně jednoslovná hodnota). Obecně však může jít o jakýkoli znak. V případě proměnné „promenna10“ je tímto znakem uzavírající hranatá závorka. Právě díky ní lze v hodnotě proměnné uvádět i mezery, jak ukazuje řádek č. 17 textového souboru.

K porušení zásady popsané v předchozím odstavci dochází pouze ve chvíli, kdy za proměnnou v šabloně následuje dvojité odřádkování. V takovém případě hodnotu proměnné v textovém souboru neohraničuje znak následující za proměnnou, nýbrž dvojice znaků, čili dvojité odřádkování. Takové proměnné lze přiřadit víceřádkovou hodnotu bez po sobě následujících odřádkování (viz řádky č. 4 a 5 textového souboru).

Nulovatelnost proměnných („@!“) a opakovaných vzorů („%!“) ovlivňuje pouze import z textového souboru. Při exportu mohou být prázdné hodnoty exportovány i do nenulovatelných proměnných („@“) a řádek v syntaxi opakovaného vzoru šablony nemusí být vůbec uveden ani v případě, že je opakování vzoru nenulovatelné („%“).

Názvy proměnných Význam proměnné v šabloně je dán jejím názvem (tj. podřetězcem za uvozujícím „@“ nebo „@!“), který ji asociuje s konkrétním atributem v databázi. Validní název proměnné má tvar „PREFIX___nazev_atributu“, přičemž „PREFIX“ je řetězec identifikující databázovou tabulku, v jejímž sche-

matu se atribut s názvem „nazev_atritutu“ nachází. Z hlediska umístění v šabloně dělíme proměnné na:

- proměnné mimo opakované vzory[47] (odkazují na atributy tabulky záznamů, užívají prefix „DATA“)
- proměnné v opakovaných vzorech – vzhledem k tomu, že v šabloně dávají smysl vždy maximálně dva opakované vzory, a to jeden pro tituly a druhý pro autory záznamu, rozlišujeme přesněji:
  - proměnné opakovaného vzoru pro tituly záznamu (odkazují na atributy tabulky titulů, užívají prefix „TITLE“)
  - proměnné opakovaného vzoru pro autory záznamu (odkazují na atributy (vazební) tabulky autorů, užívají prefix „AUTHOR“, resp. „DATA_AUTHOR“)

  Každý řádek příslušející opakovanému vzoru tedy v textovém souboru odpovídá jednomu titulu nebo autorovi záznamu. V jednom opakovaném vzoru nelze střídat proměnné titulu záznamu s proměnnými autora záznamu, ani nelze uvést opakovaný vzor bez proměnných.

Zadávání údajů o titulech a autorech záznamu působí velmi logicky právě při použití opakovaných vzorů, kdy každý řádek textového souboru představuje jeden titul nebo jednoho autora záznamu. Tento přístup však naráží na limity v oblasti přehlednosti (dlouhé a víceřádkové hodnoty titulů záznamu) a nevyhovuje ani specifickým požadavkům na export (autory požadujeme exportovat zřetězené na jediném řádku). Z tohoto důvodu jsou k dispozici zvláštní proměnné umožňující přizpůsobený import a export údajů o titulech a autorech záznamu. Tyto proměnné se vždy uvádí v částech šablony mimo opakující se vzory.

V první řadě jde o **alternativní syntaxi proměnných pro tituly**, jejíž proměnné mají tvar „TITLE_jazyktitulu___nazevatributu“. Import nebo export se týká vždy atributu „nazevatributu“, ale konkrétní titul záznamu, jehož hodnota se má použít, je určen až na základě jazyka titulu uvedeného v podřetězci „jazyktitulu“. Tento podřetězec musí být výčtovou hodnotou atributu *i_language („title“)*. (Zmíněný atribut již nelze použít v názvu proměnné na místě podřetězce „nazevatributu“.) Pokud při exportu daný záznam nedisponuje titulem v určeném jazyce, exportuje se prázdná hodnota. Výlučně jen při exportu lze pro určení jazyka použít podřetězec „orig“[48], odkazující na titul v původním jazyce záznamu[49]. Při importu titulů s použitím alternativní syntaxe vznikne pro daný záznam tolik titulů, kolik rozdílných jazyků titulu bylo použito v proměnných. Narozdíl od klasické syntaxe (použití opakovaných vzorů) nelze importovat nebo exportovat více titulů se stejným jazykem.

[47] Tj. mimo ty řádky šablony, za nimiž následuje řetězec „%“ nebo „%!“ interpretovaný jako signaliace opakování vzoru.

[48] Řetezec je určen PHP konstantou `TEMPLATE_TITLE_LANG_ORIG`.

[49] Za původní jazyk záznamu se považuje hodnota atributu *i_language_orig („data“)*.

Výhradně pro export lze použít **alternativní syntaxi proměnných pro autory**. Jedná se o sadu proměnných, jež budou při exportu nahrazeny různými druhy zřetězení autorů určeného typu. K dispozici jsou proměnné:

- „AUTHOR_typautora_1“ – zřetězení autorů ve formátu „Prijmeni1 Jmeno1, Prijmeni2 Jmeno2“
- „AUTHOR_typautora_1I“ – zřetězení autorů ve formátu „Prijmeni1 J., Prijmeni2 J.“
- „AUTHOR_typautora_2“ – zřetězení autorů ve formátu „Prijmeni1, Jmeno1; Prijmeni2, Jmeno2“
- „AUTHOR_typautora_2I“ – zřetězení autorů ve formátu „Prijmeni1, J.; Prijmeni2, J.“

Podřetězec „typautora“ se nahradí za požadovaný typ exportovaných autorů (tj. typ *Autor* nebo *Editor*). V případě typu *Editor* se na závěr zřetězení přidává zkratka „(Ed.)“, resp. „(Eds.)“.

V jedné šabloně nelze použít současně alternativní a klasickou (opakované vzory) syntaxi proměnných pro tituly nebo autory.

Výběr atributů pro import (export) Import z textového souboru je povolen pro hodnoty proměnných pevného atributu *i_public („data“)* (veřejnost záznamu) a jakýchkoli variabilních atributů. V současné implementaci je nicméně pro atributy *i_public („data“)* (veřejnost záznamu) a *i_type („data_author“)* (typ autora) definována konstantní importní hodnota, jež se pro import použije bez ohledu na hodnotu příslušné proměnné v textovém souboru (tuto proměnnou tedy obvykle vůbec neuvádíme v šabloně). Popsaný přístup eliminuje nutnost opakovaného vyplňování údajů, jejichž hodnota se u různých záznamů nemění. Pro detaily implementace viz PHP funkci `DataGetText::get()`.

Pokud některému atributu nepřísluší v šabloně použité pro import žádná proměnná, a nejde-li o atribut s implementovanou konstantní importní hodnotou (viz výše), jako importní hodnota se použije neurčená hodnota (prázdný řetězec), je-li atribut výčtového typu, a prázdná hodnota pro atributy ostatních typů.

Pro oprávnění k exportu atributů do textového souboru platí klasická pravidla oprávnění k exportu, popsaná v tabulce 5.

Korespondence šablony s parametry atributů Tvar šablony je vhodné přizpůsobit parametrům atributů, obsaženým v jejich definici v konfiguračních souborech databázového schematu (viz sekci 4.3.2).

Nenulovatelnému atributu (viz parametr *nullable*) by měla příslušet nenulovatelná proměnná (uvozena „@“). Není-li proměnná odkazující na nenulovatelný atribut v šabloně vůbec přítomna, importuje se nevalidní prázdná (nulová) hodnota a její normalizace selže. Je-li atribut výčtového typu, nenulovatelná proměnná by mu měla příslušet právě tehdy, když mezi jeho povolené výčtové

hodnoty (viz parametr *values[]*) *nepatří* prázdný řetězec. Absence proměnné v šabloně způsobí v takovém případě import hodnoty prázdného řetězce, tj. nevalidní hodnoty, a dojde k selhání normalizace.

Samozřejmostí je uvádění neveřejných atributů (viz parametr *public*) pouze v neveřejných šablonách. Při exportu podle veřejné šablony, která obsahuje nějakou proměnnou příslušnou neveřejnému atributu, vznikne v nepřihlášeném stavu chyba.

Možnost importu víceřádkových hodnot, kdy příslušnou proměnnou v šabloně zakončuje dvojité odřádkování, lze smysluplně využít pouze pro proměnné atributů interního typu *imchar*. Pro atribut typu *ichar* vznikne při importu víceřádkové hodnoty chyba normalizace.

4.3.7.3 Poznámky ke zpracování importního souboru

V případě neshody importního textového souboru a šablony rozlišujeme dvě situace. Pokud aktuální řádek šablony neobsahuje žádný pevný text, čtení daného záznamu je ukončeno s chybou a přechází se na další záznam. Pokud aktuální řádek šablony pevný text obsahuje, v textovém souboru se pro porovnání se šablonou zkusí vzít text začínající na následujícím řádku. Jestliže neshoda se šablonou trvá, berou se v textovém souboru opět texty od následujícího řádku. Proces končí nalezením shody nebo dosažením konce záznamu, kde vzniká chyba[50].

Popsaný přístup obvykle není výhodný při čtení řádků příslušných opakovanému vzoru šablony, kdy chybným zadáním některého z řádků dochází k tiché ztrátě dat (přechází se bez upozornění na další řádek). Přecházení na další řádek při neshodě se šablonou lze proto pro řádky příslušné opakovanému vzoru zapnout nebo vypnout PHP konstantou `FetchText::PATTERN_REPETITION_REDO`.

4.3.7.4 Specifika šablony exportu pro vlastní LaTeX

Šablona bibliografického exportu pro vlastní LaTeX, jejíž výchozí podobu definuje soubor `src/conf/textfile_templates/tmpl-latex-default.php`, používá obvyklou syntaxi šablony textového souboru, jak byla popsána v sekci 4.3.7.2. Některé rysy jako např. použití opakovaných vzorů nicméně uživatelská dokumentace k šabloně pro jednoduchost neuvádí.

Za účelem zvýšení uživatelského komfortu lze výhradně pro šablonu vlastního LaTeXu použít vlastní, v konfiguraci nastavené názvy proměnných (viz sekce 4.3.8).

Podrobnosti o definici a zpracování šablony a o algoritmu čtení importního souboru poskytuje soubor `doc/textfile.odt`.

[50] Číslo řádku chyby (zobrazené v přehledu chybných importních záznamů) z praktických důvodů odkazuje na řádek první neshody s šablonou, nikoli na řádek s koncem záznamu.

4.3.8 Proměnné šablony exportu pro vlastní LaTeX (template_vars_latex.php)

Bibliografický export pro vlastní LaTeX exportuje data na základě uživatelsky definované šablony, jejíž definice a následné zpracování na serveru probíhají stejným způsobem jako v případě šablony pro klasický import a export do textového souboru (viz sekci 4.3.7). Vzhledem k tomu, že u šablony pro vlastní LaTeX žádáme větší uživatelskou přívětivost (definuje ji přece sám uživatel), umožňujeme v šabloně kromě obvyklých názvů proměnných použít uživatelské názvy zavedené v souboru `src/conf/template_vars_latex.php`.

Soubor `src/conf/template_vars_latex.php` používá syntaxi jazyka PHP. Obsahuje jedinou proměnnou `$TEMPLATE_VARS`, v níž je uloženo jednorozměrné asociativní pole s definicemi mapování uživatelských názvů proměnných na obvyklé názvy. Povoleno je výhradně mapování 1:1. Přemapování proměnných se provede vždy na začátku zpracování šablony. Proměnná, která nemá definováno žádné mapování, zůstává nezměněna.

S ohledem na definované mapování se automaticky upravuje uživatelská dokumentace na stránce s editací šablony (soubor `src/latex_template.php`). Výjimku tvoří z technických důvodů proměnné odkazující na údaje titulu záznamu (proto je dobré mapování těchto proměnných vůbec nedefinovat).

4.4 Údržba

Udržování vyžadují části aplikace závislé externích systémech. V našem případě jde o externí databáze RIV (IS VaVaI) a OBD UPOL.

4.4.1 Vyhledání v externí databázi

Porovnání s externí databází začíná vždy zadáním vyhledávacích parametrů, které mají v externí databázi určit soubor dat k provedení srovnání. Generování vyhledávacího formuláře a předání vyhledávacích parametrů rozhraní externí databáze přitom nelze (tak jako u ostatních činností[51]) provést automaticky s využitím konfiguračních souborů databázového schematu – specifické vlastnosti vyhledávacího rozhraní externí databáze si vynucují zacházet s řadou vyhledávacích parametrů zvláštním způsobem, který by šlo jen obtížně v konfiguračních souborech zachytit[52] a některé atributy Agendy věd. výsledků rozhraní externí databáze neumožňuje vyhledat vůbec. Celý proces vyhledání na straně externí i vlastní databáze je tedy v kódu zapsán napevno s interními identifikátory konkrétních atributů.

Z toho vyplývá, že změny definic atributů v konfiguraci databázového schematu nelze do procesu vyhledání v ext. databázi zanést automaticky, nýbrž výhradně ruční editací kódu aplikace. Parametry vyhledávání vždy odpovídají va-

[51] Např. zobrazování údajů, vkládání údajů, importování a exportování…

[52] Někdy se vyhledává rozsah hodnot, jindy přesná hodnota; vyhledávací pole může příslušet více údajům současně; databáze RIV vyhledává pouze autory typu *Autor* atd.

riabilním atributům db. schematu se zvláštním významem (mají přidělen interní identifikátor) – jejich smazání tedy není nasnadě, ale drobná změna konfigurace vynucující úpravy pevně zapsaného kódu (např. přidání výčtové hodnoty) ano. Ruční úprava kódu je nutností i v případě přidání zcela nového vyhledávacího parametru pro vyhledání v ext. databázi.

Ještě obtížnější je údržba týkající se komunikace s externí databází. Porovnání s RIV (IS VaVaI) i OBD UPOL využívá pro požadavky na externí databázi funkcionalitu klasického webového rozhraní, u nějž se strojové dotazování nepředpokládá. Změna vyhledávacího rozhraní, jemuž parametry vyhledávání předáváme, může proto přijít nečekaně ze dne na den. (Při porovnání s databází RIV se využívá rozšířené vyhledávání veřejně přístupného rozhraní IS VaVaI[53], při porovnání s OBD UPOL pak přes rozšířený filtr neveřejného rozhraní OBD UPOL[54].)

Konkrétní požadavky na údržbu jsou vypsány ve zdrojových kódech PHP skriptů `src/compare_riv.php`, resp. `src/compare_obd.php` (generování vyhledávacího formuláře) a PHP tříd `FetchRIV`, resp. `FetchOBD` (příjem vyhledávacích parametrů, jejich úprava a přeposlání externí databázi a vyhledání ve vlastní databázi podle stejných parametrů).

Výchozí výběr vyhledávacích parametrů Ve výchozím stavu podporuje vyhledání v externí databázi (až na výjimky) všechny vyhledávací parametry z průniku vyhledávacích parametrů rozhraní externí databáze a atributů vlastní databáze.

4.4.2 Stažení exportu externí databáze

Požadavku na stažení exportu z externí databáze předchází obvykle požadavek na vyhledání, event. na přihlášení. Zaslání požadavků probíhá v PHP funkci `FetchRIV::download_data()`, resp. `FetchOBD::download_data()`.

Opět platí, že na straně rozhraní externí databáze může nastat náhlá změna, která znemožní provedení exportu až do ruční opravy aplikačního kódu (pro vyhledání a stažení exportu se nevyužívá žádné strojové rozhraní – aplikace pouze simuluje činnost prohlížeče). Obzvlášť komplikované je získání exportu z databáze OBD UPOL – podrobnosti jsou popsány v souboru `doc/export_obd.txt`. Část údržby lze provádět přímočaře změnou konstant v PHP třídách `FetchRIV` , resp. `FetchOBD` (konstanty většinou obsahují URL adresy pro požadavky na externí databázi).

4.4.3 Zpracování exportu externí databáze

Na data exportu externí databáze ukazují konf. soubory `src/conf/compare_*_cols.ini.php` a ve zvláštních případech také konstanty v PHP třídách

[53] Viz http://www.isvav.cz.
[54] Viz http://obd.upol.cz.

`DataGetRIV`, resp. `DataGetOBD`. Případná změna uspořádání exportního souboru proto implikuje nutné změny v konfiguraci. PHP třídy `DataGetRIV`, resp. `DataGetOBD`, zajišťující doručování dat z exportních souborů, generují při nedostupnosti žádaných dat výjimku.

Informaci o znakové sadě dat eportovaných z RIV (IS VaVaI) nese konstanta `DataGetRIV::IMPORT_ENCODING`. Kódování XML exportu z OBD (UPOL) je zjištěno automaticky z XML prologu.

V případě databáze RIV (IS VaVaI) lze možnou změnu uspořádání exportů indikovat podle identifikátoru datového schematu, který je umístěn v zápatí veřejného webového rozhraní[55]. Automatickou kontrolu datového schematu před stažením exportu však Agenda vědeckých výsledků nepodporuje.

4.5 Rozšiřitelnost

4.5.1 Databázové schema

Konfigurovatelnost databázového schematu (viz sekci 4.3.2) umožňuje jeho rozšíření o další atributy, stejně jako rozšíření již existujících atributů výčtového typu o nové výčtové hodnoty. Pro evidenci věd. výsledků má největší význam přidání výčtových hodnot druhu záznamu (atribut *i_type („data“)*). Rozšíření výčt. hodnot typu autora (atribut *i_type („data_author“)*) není z důvodu zvláštního významu tohoto atributu podporováno.

Přidání výčtových hodnot pro jakékoli atributy, jimž přísluší nějaký interní identifikátor, může implikovat další nutné kroky. (V praxi se to týká především atributu *i_type („data“)*.) Pro podrobnosti viz komentáře u definic příslušných atributů v souborech `src/conf/table_*.ini.php`.

4.5.2 Porovnání s externími databázemi

Vyhledání v externí databázi před zahájením porovnání lze rozšířit o další parametry pouze ruční úpravou kódu (viz sekci 4.4.1).

Zcela netriviální je pak rozšíření aplikace o porovnání s další externí databází, což vyžaduje vytvoření sady nových konfiguračních souborů a zdrojových kódů. Po vzoru porovnání s databázemi RIV (IS VaVaI) a OBD UPOL se pro novou databázi Dbnew zavádí:

- následující konfigurační soubory (srv. sekci 4.3.4):
  - `compare_dbnew_cols.ini.php`
  - `compare_dbnew_ienums.php`
  - `compare_dbnew_pair.ini.php`
- PHP skript `compare_dbnew.php` – generuje vyhledávací formulář

[55] Viz http://www.isvav.cz.

- PHP skript `compare_dbnew_script.php` – zpracuje informace z vyhledávacího formuláře, stáhne a zpracuje data pro porovnání databází a uloží je do session
- PHP třída `FetchDbnew` rozšiřující `FetchCompare` – zapouzdří sérii procesů prováděných v `compare_dbnew_script.php` (zpracování informací z vyhledávacího formuláře, vytvoření a zaslání požadavku na export z externí databáze, vyhledání dat ve vlastní databázi)
- PHP třída `DataGetDbnew` implementující `IDataGetMulti` – umožní abstrakci nad exportem externí databáze tak, aby z něj šlo žádat data způsobem, jaký určuje rozhraní `IDataGetMulti`

Dále je nutné odpovídajícím způsobem doplnit PHP funkci `FilterImportHelp::__construct()` a v případě potřeby přizpůsobit normalizaci dat z externí databáze ve třídě `FilterHelp`. Jednodušší úpravu související s proměnnou `$import_text` vyžadují PHP skripty `src/import_compare.php`, `src/import_ajax.php` a `src/import_script.php`.

Přesný postup provedení všech kroků lze odvodit podle implementace porovnání se dvěma výchozími externími databázemi RIV a OBD.

4.5.3 Bibliografické exporty

Pro přidání nového bibliografického exportu přizpůsobeného pro specifickému BibTEX citačnímu stylu (podobně jako v případě již zavedeného exportu pro styl ČSN ISO 690) postačí provést tyto kroky:

1. Vytvořit konfigurační soubor bibliografického exportu s klasickou strukturou a umístěním (viz sekci 4.3.6).
2. Doplnit volbu nového exportu do seznamu exportů. Technicky jde o přidání HTML elementu (potomka) `<option value="identifikator_exportu">Zobrazovany nazev</option>` do elementu `<select name="export">` v souborech `src/index.php` (export dostupný na titulní stránce) a `src/import_compare.php` (export v rámci porovnání s externí databází).
3. V globální PHP funkci `export()`[56] přidat do přepínače novou větev `case "identifikator_exportu":` a použít pro ni stejné příkazy jako ve větvi `case "bibtex":` s tím rozdílem, že prvním argumentem konstruktoru třídy `ExportBiblioBibtex` bude název konf. souboru vytvořeného v kroku 1.

V případě záměru zavést export s odlišnou strukturou, než má obvyklý seznam literatury BibTEXu, je nutné nejprve naprogramovat PHP třídu rozšiřující `ExportBiblio` a tuto strukturu v ní definovat. Pak již lze postupovat podle instrukcí výše s tím, že v kroku 3 se namísto třídy `ExportBiblioBibtex` použije

[56] Viz `src/load/functions.php`.

nově naprogramovaná třída. Tímto způsobem byl zaveden bibliografický export do formátu DBLP XML (třída `ExportBiblioDBLP`).

4.5.4 Šablony textového souboru

Pro importy a exporty do textového souboru lze dodefinovat vlastní šablony. Veškeré informace obsahuje sekce 4.3.7.

4.6 Nástroje správce

Tato sekce shrnuje funkcionalitu webového rozhraní aplikace dostupnou výhradně uživateli s úrovní oprávnění *administrátor*.

4.6.1 Aktualizace databázového schematu

Volbou z nabídky *Nástroje správce – Aktualizace databázového schematu* se otevře rozhraní umožňující automatickou úpravu databázového schematu na základě změn nalezených v konfiguračních souborech db. schematu jednotlivých tabulek (srv. sekci 4.3.2).

Podporováno je vytvoření nových databázových atributů a doplnění dalších výčtových hodnot pro již existující atributy výčtového typu. Jakékoli další změny konfiguračních souborů db. schematu, mají-li dopad na definici atributů v databázi[57], musí být do databázového schematu přenesy ručně, tj. provedením vlastního SQL příkazu.

4.6.2 Správa uživatelů

Volba z nabídky *Nástroje správce – Správa uživatelů* zobrazí přehled všech uživatelů aplikace s možností jejich smazání, změny hesla a s možností vytvoření nového uživatele. Příkaz pro smazání uživatele je dostupný pouze v případě, že daný uživatel není vlastníkem žádného záznamu nebo autora. (Smazání uživatele tedy musí předcházet převedení všech jím vlastněných záznamů a autorů na jiného vlastníka, což lze hromadně provést pouze ručním SQL příkazem.) Uživatele s úrovní oprávnění *administrátor* („admin“) nelze smazat nikdy.

Uváděný email uživatele slouží administrátorovi ke kontaktním účelům. Automatické rozesílání elektronické pošty neprobíhá.

[57] Jde o změny parametrů *type*, *nullable*, *length* a *values[]* v definicích atributů v konf. souborech.

Závěr

Implementovaná aplikace umožňuje díky podpoře úpravy a rozšíření sady evidovaných údajů evidenci obecně jakýchkoli druhů vědeckých výsledků. Zvláštní důraz je přitom kladen na výsledky publikačního rázu. Kromě interaktivního prohlížení může uživatel záznamy hromadně exportovat, a to v různé formě, závislé na účelu exportu. Alternativu k interaktivnímu vkládání záznamů představuje dávkový import z textového souboru. Funkcionalita porovnání dat s příbuznou (externí) databází umožňuje pohodlným způsobem detekovat rozdílná nebo ve vlastní databázi chybějící data a provést jejich hromadný import. Procesy exportu a importu disponují rozsáhlými možnostmi konfigurace a rozšiřitelnosti.

Jako stinnou stránku provozování aplikace lze vnímat nevyhnutelné požadavky na údržbu, generované těmi částmi aplikace, jež zajišťují komunikaci s rozhraním externích databází nebo zpracování jejich exportů.

Do oblasti potenciálního vylepšení funkčnosti spadá flexibilnější koncepce systému oprávnění (v současnosti disponuje právem editace záznamu pouze jeho vlastník a administrátor), podpora fulltextového vyhledávání nebo také rozšíření nabídky externích databází, s nimiž lze provádět porovnání dat.

Conclusions

Thanks to the support of adjustment and extensibility of database schema, implemented application provides record keeping of research results of any kind. Specific emphasize is put on publication results. Apart from interactive browsing, user is allowed to perform various types of export. Each type of export serves a particular purpose. As an alternative to the interactive record insertion can be used batch import from text file. Functionality of data comparison between local and related database of research results enables user to detect different or missing data and successively execute a batch import. Processes of export and import have great abilities in terms of configuration and extensibility.

The application inevitably requires maintenance of modules implementing communication with interfaces of related (external) databases or processing of their exported data.

To the area of possible improvements belong more flexible concept of permission system (presently only owner and administrator have parmission to edit a record), support of fulltext search or adding a new related database for the functionality of database data comparison.

A Výchozí evidované údaje

Konfigurační soubory databázového schematu[58] a v návaznosti na ně i ostatní konf. soubory předpokládají ve výchozím stavu rozsáhlou evidenci zejména publikačních vědeckých výsledků. Tato sekce popisuje způsob, jakým byl výběr výchozích evidovaných údajů proveden.

Výběr údajů evidovaných pro tituly a autory vědeckých výsledků není nutné komentovat. Bude pravděpodobně stejný bez ohledu na to, kde je aplikace aktuálně nasazena. Výběr údajů týkajících se záznamů o vědeckých výsledcích jako takových už ale závisí na druhu vědeckých výsledků, které se mají evidovat.

Ve výchozím stavu se předpokládá evidence těchto druhů vědeckých výsledků (v závorce uvedena odpovídající výčtová hodnota atributu *i_type („data“)*):

- článek v časopisu obsaženém v databázi Web of Science (Jimp)
- článek v časopisu obsaženém v databázi SCOPUS (JSC)
- článek v časopisu obsaženém v Seznamu neimpaktovaných recenzovaných periodik (Jrec)
- článek v časopisu obsaženém v databázi ERIH (Jneimp)
- článek bez rozlišení poddruhu (J)
- kniha (B)
- kapitola v knize (C)
- příspěvek ve sborníku (D)
- software (R)[59]

Evidence druhu příspěvek ve sborníku konference (D) si například vynucuje evidenci údajů název konferece, místo konání konference, datum zahájení konference aj., které by jinak v databázovém schematu být nemusely. Naopak údaje jako rok uplatnění nebo URL odkaz na online verzi výsledku musí být evidovaný pro všechny druhy výsledků.

Korespondence s bibliografickými formáty Jelikož aplikace umožňuje export informací o věd. výsledcích do formátů seznamů literatury, evidujeme pokud možno také všechny údaje, které BibTEX standard [4] definuje jako povinné pro *BibTEX typy* příslušné evidovaným druhům výsledků[60].

[58] Viz sekci 4.3.2.

[59] Software je ve výčtu druhů jediným nepublikačním druhem výsledku.

[60] Mapování druhů výsledků na *BibTEX typy* definují konf. soubory bibliografických exportů (viz sekci 4.3.6).

Návaznost na externí databáze Pro dobrou výchozí konfiguraci porovnání s externími databázemi bylo nezbytné databázové schema srovnat a doplnit vzhledem ke schematům těchto ext. databází. Rozhodujícím přitom nemohlo být vlastní schema ext. databáze, nýbrž údaje dostupné v získávaném exportu (pro databázi RIV je to **DBF** export, pro OBD UPOL **XML** export). Přehled o údajích dostupných v exportech, stejně jako o jejich výchozím mapování na údaje ve schematu Agendy vědeckých výsledků[61], obsahuje soubor `doc/databazove_schema.ods`.

Datové typy evidovaných údajů

Interní datové typy[62] atributů příslušných evidovaným údajům byly zvoleny podle praktické potřeby a na základě znalostí o datových typech ekvivalentních atributů v externích databázích (viz opět `doc/databazove_schema.ods`).

Co se týká výchozí sady výčtových hodnot atributů interního dat. typu *ienum*, obvykle jde o výsek hodnot z některého číselníku na webu Sekce místopředsedy vlády pro vědu, výzkum a inovace http://www.vyzkum.cz[63]. Přehled výchozího mapování výčtových hodnot na hodnoty výčtových údajů v externích databázích[64] obsahuje soubor `doc/databazove_schema.ods`.

B Použité značení a pojmy

- *záznam*, *titul*, *autor* – názvy základních databázových entit. Jejich význam vysvětluje sekce 2.2.1.
- *i_interni_identifikator („kod_tabulky“)* – označení atributu s interním identifikátorem *i_interni_identifikator*, náležejícího do tabulky s kódem *kod_tabulky*. Tedy např. *i_public („data“)* označuje atribut veřejnost tabulky záznamů. Podrobnosti o interních identifikátorech obsahuje sekce 4.3.2. Kódy tabulek vysvětluje tabulka 1.
- Uváděné prvky jazyka PHP jsou vyznačeny v souladu se syntaxí tohoto jazyka:
  - `$promenna`
  - `funkce()`
  - `KONSTANTA`
  - `Trida::$clenska_promenna`
  - `Trida::clenska_funkce()`
  - `Trida::KONSTANTA_TRIDY`

[61] Srv. sekci 4.3.4.1.
[62] Viz tabulku 6.
[63] Více viz komentáře v konfiguračních souborech.
[64] Srv. sekci 4.3.4.2.

C Obsah přiloženého CD

`bin/`
: Kompletní adresářová struktura webové aplikace (v ZIP archivu), obsahující soubory pro přenesení na webový server. (Bližší popis adresářové struktury obsahuje sekce 2.3.)

`doc/`
: Text bakalářské práce ve formátu PDF a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu). Dále je obsaženo několik souborů přiložené dokumentace, která doplňuje části dokumentace v textu práce.

`src/`
: Kompletní zdrojové texty webové aplikace se všemi zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření adresářové struktury na webovém serveru. (Bližší popis adresářové struktury obsahuje sekce 2.3.)

`readme.txt`
: Instrukce pro nasazení webové aplikace na webový server a webová adresa, na které je aplikace nasazena pro účel testování při tvorbě posudků práce a pro účel obhajoby práce.

Literatura

[1] Sekce místopředsedy vlády pro vědu, výzkum a inovace. *Koncepce Informačního systému výzkumu, experimentálního vývoje a inovací na období 2016 až 2020*. 2016. Dostupný z: ⟨http://www.vyzkum.cz⟩.

[2] Mašláň, Miroslav. *Evidence publikační činnosti na Univerzitě Palackého v Olomouci: Směrnice rektora UP č. B3-10/6*. 2010.

[3] Satrapa, Pavel. *LaTeX pro pragmatiky*. 2011. Dostupný také z: ⟨http://www.nti.tul.cz/~satrapa/docs/latex/latex-pro-pragmatiky.pdf⟩.

[4] Patashnik, Oren. *BibTeXing*. 1988. Dostupný z: ⟨http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf⟩.

[5] Lehman, Philipp; Kime, Philip; Boruvka, Audrey; Wright, Joseph. *The BibLaTeX Package: Programmable Bibliographies and Citations*. 2016. Dostupný z: ⟨http://mirrors.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf⟩.

[6] Ley, Michael. DBLP - Some Lessons Learned. *PVLDB*. 2009, roč. 2, č. 2, s. 1493–1500. Dostupný také z: ⟨http://www.vldb.org/pvldb/2/vldb09-98.pdf⟩.

[7] Achour, Mehdi; Betz, Friedhelm; Dovgal, Antony aj. *PHP Manual*. 2016. Dostupný z: ⟨http://php.net/manual⟩.