České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Integrace externích bibliografických zdrojů do Vokabuláře webového

Tomáš Hrabáček

Vedoucí práce: Ing. Vladimír Pokorný
Studijní program: Softwarové inženýrství a technologie
Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hrabáček** Jméno: **Tomáš** Osobní číslo: **466354**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Integrace externích bibliografických zdrojů do Vokabuláře webového

Název bakalářské práce anglicky:

Integration of external bibliographic resources into the Vokabulář webový

Pokyny pro vypracování:

Proveďte analýzu dostupných externích bibliografických zdrojů použitelných pro portál Vokabulář webový. Dostupná data důsledně analyzujte z hlediska integrace do existujícího projektu Vokabulář webový. Navrhněte a implementujte nástroj pro integraci bibliografických dat z externích zdrojů. Implementaci proveďte za využití technologií použitých v portálu Vokabulář webový – ASP.NET Core. Integraci důsledně otestujte a dokumentujte.

Seznam doporučené literatury:

THANOS, Constatino. Researchand Advanced Technology for DigitalLibraries:6th European Conference, ECDL 2002, Rome, Italy, September16-18,2002. Proceedings Springer Berlin Heidelberg,2003. ISBN 978-3-540-45747-3.
MICHAEL, J. James a Maark HINNEBUSCH. From A to Z39.50: A Networking Primer (Supplement to computers in libraries). Mecklermedia Corporation, 1995. ISBN-13: 978-0887367663.
Vokabulář webový, dostupný z: https://vokabular.ujc.cas.cz/
TROELSEN, Andrew a PhilipJAPIKSE. Pro C#7: With.NET and.NET Core. Apress2017. ISBN:978-1484230176.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Vladimír Pokorný, katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **24.01.2019** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Vladimír Pokorný podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce, Ing. Vladimíru Pokornému, za cenné rady, které pomohly k jejímu dokončení. Dále děkuji rodině za podporu v průběhu studia i při vypracování této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2019

Abstrakt

Tato práce se zabývá analýzou, návrhem a implementací služby pro Vokabulář webový, která do portálu importuje metadata o dílech z externích bibliografických zdrojů. Služba je navržena tak, aby ji bylo v budoucnu možno snadno rozšířit. Pro komunikaci s externími bibliografickými zdroji byl implementován protokol OAI-PMH. Bibliografický formát, který služba zpracovává se nazývá MARC 21.

Klíčová slova: Vokabulář webový, OAI-PMH, Z39.50, MARC 21, MARCXML

Vedoucí práce: Ing. Vladimír Pokorný

Abstract

This thesis deals with the analysis, design and implementation of the Vokabulář webový service, which imports metadata about literaly works from external bibliographical resources to the portal. The service is designed for easy extension in the future. For communication with external bibliographical resources is implemented protocol OAI-PMH and bibliographical format, which is processed by service is called MARC 21.

Keywords: Vokabulář webový, OAI-PMH, Z39.50, MARC 21, MARCXML

Title translation: Integration of external bibliographic resources into the Vokabulář webový

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Tato práce se zabývá problematikou získávání dat z různých webových rozhraní, jejich následného zpracovávání, uložení a aktualizace. V případě Vokabuláře webového se jedná o získávání metadat o dílech pomocí protokolu Open Archives Initiative Protocol for Metadata Harvesting (dále jen OAI-PMH), případně pomocí protokolu Z39.50.

1.1 O Vokabuláři webovém

Vokabulář webový je webový portál, který zpřístupňuje textové, obrazové a zvukové zdroje a slouží k poznání historické češtiny. Tvůrcem a provozovatelem Vokabuláře webového je oddělení vývoje jazyka Ústavu pro jazyk český AV ČR, v. v. i. (dále jen ÚJČ). Vokabulář webový je postupně doplňován o digitalizované zdroje týkající se historické češtiny. Účelem Vokabuláře webového je zpřístupnit historickou češtinu v elektronické podobě zájemcům z řad široké veřejnosti i odborníků [1].

Od roku 2010 na vývoji Vokabuláře webového spolupracuje České vysoké učení technické. První spoluprací byl projekt pod názvem Informační technologie ve službách jazykového kulturního bohatství (IT Jakub). Tento projekt měl za cíl zpřístupnit Vokabulář webový na různé platformy, např. počítače, mobilní telefony, audio přehrávače. Současně byl vytvořen software a metodiky pro převod nehmotného kulturního dědictví do elektronicky čitelné podoby. V poslední části vznikly nástroje pro zapojení materiálů z Vokabuláře webového do moderních výukových metod na různých typech škol. Projekt byl ukončen v roce 2015 [2].

Další spolupráce probíhá v rámci projektu Výzkumná infrastruktura pro diachronní bohemistiku (akronym RIDICS). Tento projekt má trvání od roku 2016 do roku 2019, cílem je vytvořit dva portály umožňující výzkum

v oblasti diachronní bohemistiky a souvisejících oborů. Prvním portálem bude badatelský webový portál, který bude sloužit odborníkům pro výzkum a zpřístupnění odborných různorodých zdrojů [3].

Druhým portálem bude komunitní portál, jenž bude přístupný kromě odborníků také studentům a laické veřejnosti. Portál bude umožňovat uživatelům sdílet výsledky jejich výzkumu a diskutovat o odborných tématech [3].

1.2 Cíl práce

Cílem této práce je analyzovat požadavky a možnosti integrace externích bibliografických databází do Vokabuláře webového, což obsahuje analýzu dostupných externích bibliografických databází a rozhraní, které podporují tyto databáze, přes které lze data získat. Dále je nutno provést analýzu datových formátů, ve kterých analyzovaná rozhraní poskytují data. Kromě analytické práce bude také navržena možná podoba implementace a zároveň bude vytvořen prototyp v podobě webové aplikace, která bude zahrnovat získání dat z jedné vybrané externí bibliografické databáze a následné zobrazení získaných dat v přehledné formě uživateli.

1.3 Motivace

Hlavním důvodem pro rozšíření Vokabuláře webového o integraci externích bibliografických zdrojů je soustředit dostupné prameny a záznamy o historické češtině na jedno místo. Díky tomu budou moci uživatelé Vokabuláře webového jednoduše uvádět, ze kterých děl čerpali, případně jaká díla mají podobný charakter obsahu či podobné jiné specifikum.

Kapitola 2

Analýza

Tato kapitola je věnována popisu struktury Vokabuláře webového a technologiím, které se v tomto projektu využívají. Dále je věnována definování požadavků na integraci externích bibliografických zdrojů do Vokabuláře webového, analyzování dostupných externích bibliografických databází a podrobnějšímu popisu API, které tyto databáze poskytují, a zároveň popisu knihovnických formátů pro výměnu dat, které jednotlivé API poskytují.

2.1 Struktura Vokabuláře webového

Vokabulář webový je webová aplikace, která je postavená na technologii .NET. Aplikace je rozdělena do několika samostatných služeb, které obstarávají jednotlivé požadavky uživatelů. Obrázek 2.1 vyobrazuje rozdělení Vokabuláře webového na jednotlivé služby.

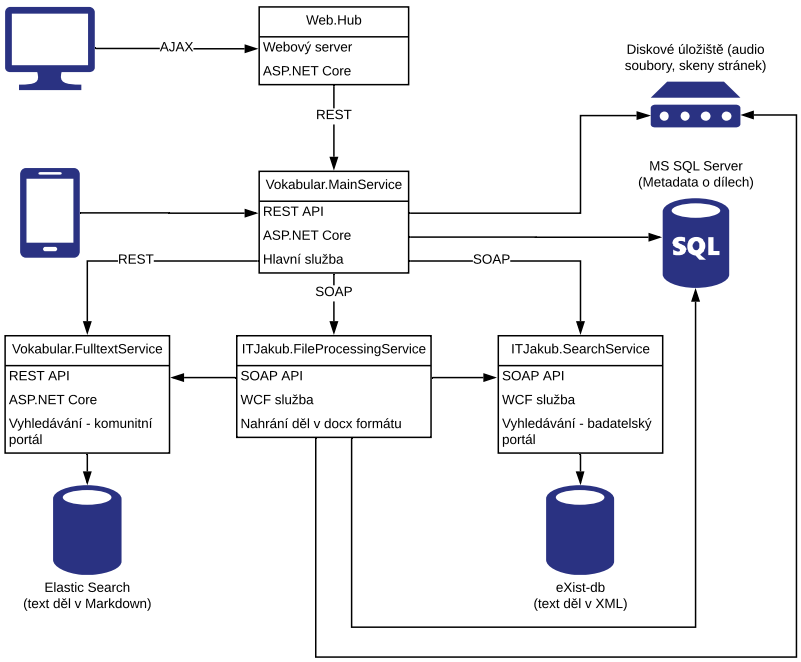
Web.Hub zde slouží jako webový server, který obsluhuje dotazy z webových prohlížečů. Využívá architektury Model-View-Controller, tudíž příchozí požadavek zpracovává příslušný controller. Controllery přijaté požadavky odesílají na hlavní službu - *Vokabular.MainService*. Hlavní služba dotaz zpracuje, případně odešle požadavek dále - specializované službě, která ho zpracuje, např. získá požadovaná data z databáze, které má dostupné. Následně se tato data vrací z hlavní služby do webového serveru ve formátu JSON, kde jsou zpracována a vrácena webovému prohlížeči.

Hlavní služba Vokabuláře webového vystavuje REST API, které umožňuje jiným aplikacím komunikovat skrze toto rozhraní přímo s hlavní službou. Jako příklad může být mobilní aplikace.

Vokabulář obsahuje 2 databáze ve kterých jsou uloženy texty děl. První databázi eXist-db využívá služba *ITJakub.SearchService*. Ta umožňuje fulltextové vyhledávání na badatelském portálu. Elasticsearch je druhou databází,

která uchovává texty děl. Je využívána službou *Vokabular.FulltextService* a umožňuje vyhledávání na komunitním portálu.

Služba *ITJakub.FileProcessingService* slouží pro zpracovávání nahraných děl. Nahraná díla zpracuje, uloží metadata do MS SQL databáze a text předá jedné ze služeb *Vokabular.FulltextService* a *ITJakub.SearchService*, která si je uloží do svých databází.



Obrázek 2.1: Struktura Vokabuláře webového

2.2 Použité technologie

Integrace externích bibliografických databází bude probíhat nad již existujícím projektem, tudíž tato kapitola se zabývá analýzou technologií, jenž se aktuálně (stav ze dne 25. 1. 2019) v projektu využívají. Analýza se týká zejména částí *Web.Hub* a *MainService*.

2.2.1 ASP.NET Core

Serverová část Vokabuláře webového využívá framework ASP.NET Core. Tento framework je součástí platformy .NET Core, která slouží mimo jiné i pro tvorbu webových aplikací [4].

Projekt Vokabulář webový využívá architektury Model-View-Controller, kterou podporuje i webový framework ASP.NET Core [4]. Architektura MVC je založena na oddělení dat a jejich prezentace. Model je část, která představuje data a logiku aplikace. View neboli pohled zajišťuje prezentaci dat. Komunikaci mezi modelem a pohledy zajišťuje část zvaná controller [5].

Části model a controller jsou psány v jazyce C#. Pohledy jsou tvořeny HTML šablonami, ve kterých je využita syntaxe Razor. Ta umožňuje použití jazyka C# v HTML šablonách [6].

Dále je v celém Vokabuláři webovém využito návrhového vzoru Inversion of Control (dále jen IoC), jehož základním principem je otočení řízení vykonávání programu [7]. V tomto případě IoC framework volá kód programátora a řídí jeho chod. Kód vyžaduje pro svou práci další třídy, ty mu dodá IoC kontejner například pomocí konstruktoru. IoC se zároveň stará o kompletní životní cyklus tříd.

2.2.2 Microsoft SQL Server

Ve Vokabuláři webovém se v současném stavu využívají 3 databázové systémy. Prvním využívaným je relační databázový systém Microsoft SQL Server [8]. V této databázi jsou uloženy například uživatelé, skupiny, jejich práva a mnoho dalších informací potřebných pro chod portálu. V této práci budou využívaný převážně tabulky jenž obsahují metadata o dílech, které jsou k dispozici ve Vokabuláři webovém.

Pro uložení děl samotných se využívají NoSQL databáze eXist-db a Elasticsearch. Tyto databáze nejsou v bakalářské práci využívány z důvodu importu pouze metadat o dílech, ne děl samotných.

2.2.3 Typescript

Vygenerované HTML šablony jsou na straně klienta doplněny o TypeScript. Jedná se o nadstavbu JavaScriptu, oproti němu TypeScript obsahuje například statickou typovou kontrolu, podporuje genericitu, výčtové typy [9].

2.2.4 LESS

LESS je nadstavbou jazyka CSS, jedná se o preprocesor, který kód napsaný v jazyce LESS kompiluje do klasického CSS [10]. Slouží pro úpravu stylů jednotlivých stránek Vokabuláře webového. Mezi jeho hlavní výhody oproti CSS patří podpora proměnných a funkcí, dále umožňuje zápis vnořených pravidel a další možnosti, které usnadňují a zpřehledňují psaní kódu [10].

2.3 Požadavky

Na základě komunikace s odborníky z ÚJČ a analýzy potřeb ÚJČ, byly sepsány požadavky pro vytvoření služby importující externí bibliografické záznamy do Vokabuláře webového.

2.3.1 Funkční požadavky

1. Systém musí umožnit administrátorům Vokabuláře webového importovat externí bibliografické záznamy z digitálních archivů, jenž jsou dostupné přes webové rozhraní pomocí protokolu OAI-PMH (dále jen zdroj), tj., umí zpracovat přijatá data ve formátu MARC XML a uložit je do relační databáze. K tomu by mělo sloužit administrační rozhraní, které umožní uživatelům následující:
   - Uživatel může přidat zdroj, ze kterého se budou importovat externí bibliografické záznamy. Pro uložení zdroje jsou nutné tyto údaje:
     - Název zdroje.
     - Typ zdroje.
   - Uživatel může upravit konfiguraci zdroje, ze kterého se budou importovat externí bibliografické záznamy.
   - Uživatel může odebrat zdroj, ze kterého se budou importovat externí bibliografické záznamy.
   - Uživatel může ručně spustit import externích bibliografických záznamů a vybrat, z jakých zdrojů bude import proveden.
   - Systém umožní uživateli zobrazení informací o posledním importu z daného zdroje. Zejména to jsou tyto informace:
     - Kdy byl naposledy proveden import.
     - Jaký uživatel provedl poslední import.
     - Kolik nových záznamů přibylo při posledním importu.
     - Kolik záznamů bylo aktualizováno při posledním importu.

- Kolik záznamů bylo importováno za celou dobu existence tohoto zdroje.

2. Importované externí bibliografické záznamy se budou zobrazovat ve Vokabuláři webovém v kategorii Bibliografie.
3. Importované položky budou rozšířeny o odkaz na originální záznam na webových stránkách instituce, ze které byly získány.

2.3.2 Nefunkční požadavky

1. Služba pro komunikaci pomocí protokolu OAI-PMH bude implementována jako samostatná knihovna, která bude následně integrována do Vokabuláře webového.
2. Služba pro konverzi dat bude implementována jako samostatná knihovna, která bude následně integrována do Vokabuláře webového.
3. Služba bude navržena takovým způsobem, aby bylo umožněno přidávání dalších komunikačních klientů a konverzních nástrojů pro zpracování přijatých dat.
4. Pro implementaci služby budou použity technologie, které využívá Vokabulář webový v aktuálním stavu (ze dne 25. 1. 2019).

2.4 Analýza externích bibliografických databází

Všechny zdroje, které dodal ÚJČ, jsou bibliografické databáze, jenž pochází z výzkumných ústavů spadajících pod Akademii věd České republiky (dále jen AV ČR). Obsah zdrojů se nemění příliš často a pro potřeby ÚJČ bylo dohodnuto manuální importování skrze administrační rozhraní Vokabuláře webového, které budou provádět pověření pracovníci ÚJČ na základě své potřeby. Následuje přehled dodaných bibliografických databází.

Bibliografie české lingvistiky

- Instituce: Ústav pro jazyk český AV ČR.
- Webové stránky instituce: `https://bibliografie.ujc.cas.cz`.
- Protokol pro přístup: OAI-PMH.
- Formát dat: MARCXML, DC.
- Přístup: OAI-PMH protokol bude zpřístupněn v budoucnu dle dostupných informací.

Bibliografie dějin Českých zemí

- Instituce: Historický ústav AV ČR.
- Webové stránky instituce: `https://biblio.hiu.cas.cz/`.
- Protokol pro přístup: OAI-PMH.
- Formát dat: MARCXML, DC.
- Přístup: `http://biblio.hiu.cas.cz/api/oai` (set: „cpk-huav“).

Česká literární bibliografie

- Instituce: Ústav pro českou literaturu AV ČR.
- Webové stránky instituce: `https://clb.ucl.cas.cz/cs-cz/`.
- Bibliografie české literární vědy (od roku 1945).
- Protokol pro přístup: OAI-PMH, Z39.50.
- Formát dat: MARCXML, DC.
- Přístup: OAI-PMH protokol bude zpřístupněn v budoucnu dle dostupných informací.

2.5 Analýza protokolů

Dodané bibliografické databáze využívají specifické protokoly pro poskytování dat, jejich struktura a použití je podrobněji vysvětleno v této sekci.

2.5.1 Open Archives Initiative Protocol for Metadata Harvesting

Open Archives Initiative Protocol for Metadata Harvesting je protokol pro komunikaci mezi informačními systémy, založený na principu sklízení dokumentových metadat[1], jež vyvinula organizace Open Archives Initiative (dále jen OAI) za účelem zlepšení a usnadnění interoperability mezi jednotlivými informačními systémy [11].

[1] Metadata jsou data, která poskytují informace o jiných datech.

Historie protokolu

V 90. letech se začaly rozšiřovat počty archivů elektronických tisků. Rozhraní pro vkládání dokumentů byla značně odlišná pro jednotlivé archivy. Odlišné rozhraní výrazně ztěžovalo možnosti automatizace procesu přejímání záznamů z jiných repozitářů. Z toho důvodu vznikl protokol OAI-PMH, jeho cílem je usnadnit sklízení záznamů z digitálních repozitářů [12].

První představení protokolu proběhlo roku 2001 pod názvem OAI-PMH verze 1.0, tato i následně vydaná verze 1.1 byly pouze experimentální, proto se nadále budeme zabývat pouze verzí protokolu 2.0, jenž byla vydána v roce 2002. Verze 2.0 není zpětně kompatibilní s předchozími verzemi [13].

Využití protokolu

Ve světě je hojně využíván, má širokou podporu v různých typech systémů, např. systém EThOS vyvinutý institucí British Library nebo Mercury: Metadata Search System podporovaný společností NASA [14, 15].

V České republice můžeme nalézt využití například u Souborného katalogu České republiky (dále jen SK ČR), jejž zřizuje Národní knihovna České republiky. SK ČR umožňuje knihovnám získávání záznamů pomocí protokolu OAI-PMH a zároveň stejnou cestou i přispívání do SK ČR [16].

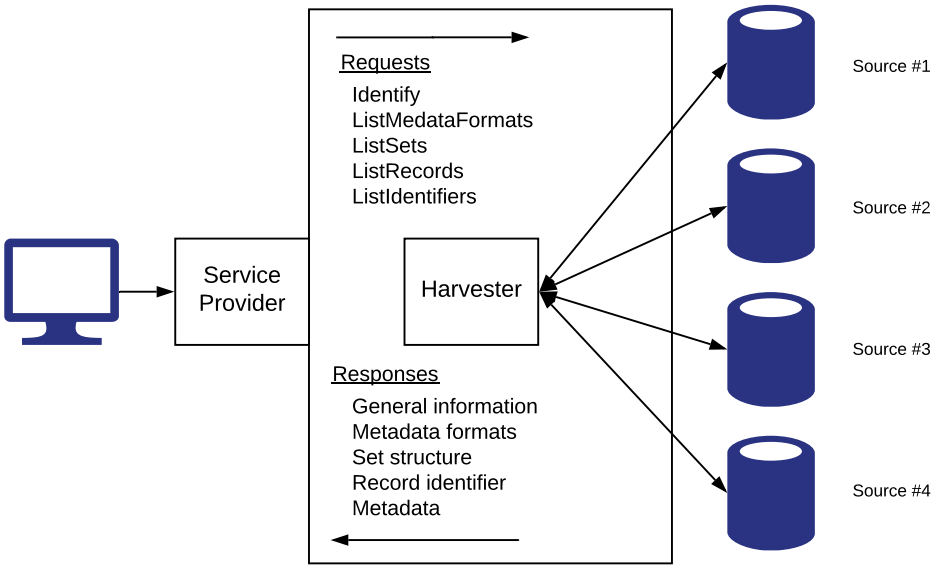
Podstata protokolu

Komunikace pomocí protokolu OAI-PMH je založena na principu architektury klient server, kde digitální repozitář je v roli serveru a sběrač v roli klienta (viz obrázek 2.2). OAI-PMH představuje webové API, které využívá metod internetového protokolu HTTP a to především jeho dotazovacích metod GET a POST. Pomocí těchto metod se sběrač dotazuje na URL adresy repozitáře a tím ve výsledku volá metody, které poskytuje API. Repozitář vrací data ve formátu XML [2] [11].

Struktura protokolu

Hlavním kritériem pro výběr volané metody webového API jsou protokolem definovaná klíčová slova (*Identify, ListMetadataFormats, ListSets, ListIdenti-*

[2] Schéma XML je určeno pomocí XSD, které se nachází na adrese `http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd`.



Obrázek 2.2: Struktura modelu protokolu OAI-PMH

fiers, ListRecords, GetRecord). V URL tomu přísluší argument *verb*, který se musí nacházet ve všech dotazech.

Klíčové slovo *Identify* v odpovědi vrací údaje o daném repozitáři. Například obecné údaje - název repozitáře, popis, email na správce repozitáře, ale i technické, jako jsou verze implementovaného protokolu OAI-PMH, granularita časové značky, která je přidávána k záznamům za účelem zaznamenání poslední modifikace záznamu (možné jsou jen dvě verze: datum nebo datum a čas s přesností na sekundy a vymezením časového pásma). Dále je zde hodnota první časové značky, jenž udává spodní hranici, pod kterou se nesmí nacházet časové značky záznamů. Poslední informací v odpovědi je úroveň podpory smazaných záznamů.

Specifikace OAI-PMH určuje 3 úrovně podpory smazaných záznamů:

- *no* - repozitář neudržuje informaci, jestli je záznam smazán,
- *persistent* - repozitář udržuje informaci, jestli je záznam smazán,
- *transient* - repozitář udržuje informaci, jestli je záznam smazán, ale negarantuje, že tato informace je aktuální.

I když je záznam smazaný a repozitář udržuje tuto informaci, stále se bude záznam vyskytovat v odpovědích na dotazy *ListIdentifiers* a *ListRecords*, ale v hlavičce záznamu přibude atribut *status*, který bude mít hodnotu *deleted*. V případě dotazu s klíčovým slovem *GetRecord* bude vrácena v odpovědi chyba typu *idDoesNotExist*.

Protokol umožňuje dva způsoby získávání záznamů. Prvním způsobem je získávání jednotlivých záznamů pomocí klíčového slova *GetRecord*. Pro tento způsob získávání záznamů je ale nutné znát identifikátor[3] hledaného záznamu. Ten lze získat pomocí klíčového slova *ListIdentifiers*, to vrací seznam identifikátorů, které jsou dostupné v tomto repozitáři, seznam lze například omezit pomocí zadání níže zmíněného setu. Druhým způsobem získání záznamů je využití klíčového slova *ListRecords*, odpovědí na tento dotaz je seznam záznamů, které se nachází v repozitáři. Stejně jako u získání seznamu identifikátorů, lze i zde omezit výsledné záznamy pomocí setů.

Pro klíčová slova *ListIndetifiers* a *ListRecords* můžeme zadat hodnoty pro nepovinné argumenty *from* a *until*. Tyto argumenty určují rozsah, ve kterém se musí nacházet časová značka záznamů (ta se mění v případě přidání, aktualizace nebo smazání záznamu z repozitáře). Tento rozsah je inkluzivní, tudíž argument *from* musí být interpretován jako „větší nebo rovno“ a argument *until* jako „menší nebo rovno“ [11]. Tento způsob tedy umožňuje tzv. inkrementální sklízení dat - sklízení pouze záznamů, které byly modifikovány od posledního sklízení.

Hierarchie záznamů

Protokol poskytuje podporu pro tzv. sety. Sety jsou organizačním prvkem protokolu OAI-PMH a představují možnost, jak klientům zjednodušit práci při získávání dat - vystavení setu pouze s určitým druhem záznamů, který je zajímá. Podporovány jsou dva druhy setů:

- Jednoduché – tento princip můžeme přirovnat k tagům či klíčovým slovům, které se využívají například na webových publicistických portálech k označení příspěvků stejného tématu či kategorie.
- Hierarchické – set může obsahovat další sety. Tento způsob se dá aplikovat na reprezentaci struktury vysoké školy. Vysoká škola má své fakulty a fakulty mají jednotlivé katedry.

Jednotlivé záznamy mohou být řazeny do setů, jeden záznam může příslušet libovolnému počtu setů. Seznam dostupných setů získáme pomocí klíčového slova *ListSets*.

Formáty metadat

U klíčových slov, která mají vracet identifikátory (*ListIdentifiers*) či přímo již záznamy (*ListRecords, GetRecord*), je nutné také specifikovat hodnotu argumentu *metadataPrefix*, což je určení v jaké XML schématu požadujeme dotazované záznamy vrátit. Záznamy mohu být v různých schématech, OAI-PMH specifikace pouze zavádí povinnost standardu nekvalifikovaný Dublin

[3] Záznamy, které repozitáře poskytují musí mít jednoznačné identifikátory.

Core [11]. Nekvalifikovaný Dublin Core je standard pro popis metadat digitálních objektů, ve verzi 1.1 je určen sadou 15 základních prvků, z nichž není ani jeden povinný [22]. V našem případě budou záznamy poskytovány ještě ve schématu MARC XML. Pokud chceme zjistit, jaká schémata daný repozitář podporuje, položíme dotaz s klíčovým slovem *ListMetadataFormats*.

Propagace chyb

V případě zachycení chyby nebo výjimky při zpracování dotazu vrací vlastní chybové stavy, které jsou odlišné od HTTP stavových kódů. Docíleno toho je přidáním jednoho či více elementů do těla odpovědi v závislosti na počtu zachycených chyb a výjimek. Specifikace protokolu OAI-PMH definuje sadu 8 chyb. Každá jednotlivá chyba má svůj kód a seznam klíčových slov, u kterých se může vyskytovat.

Ukázka komunikace

Na obrázku 2.3 se nachází příklad odpovědi na dotaz pomocí protokolu OAI-PMH s klíčovým slovem *Identify*, na rozhraní serveru Historického ústavu AV ČR, který podporuje komunikaci pomocí protokolu OAI-PMH. Dotaz vypadá následovně `https://biblio.hiu.cas.cz/api/oai?verb=Identify`. Klíčové slovo *Identify* poskytuje informace o repozitáři. Informace o dotazu lze nalézt v elementu *request*, v jeho atributu *verb* vidíme použité klíčové slovo *Identify*. Informace vrácené v odpovědi, a celkově celá odpověď se řídí podle XML schématu definovaného XSD souborem[4], vydaným organizací OAI.

Pro klíčová slova *ListSets, ListIdentifiers, ListRecords* je dostupný speciální argument *resumptionToken*. Používá se v případě, že seznam položek k vrácení je příliš dlouhý, a tudíž dojde k jeho rozdělení a vrácení pouze části seznamu spolu s elementem *resumptionToken*. Při použití hodnoty elementu *resumptionToken* jako hodnoty argumentu při dotazu, bude vrácena další část seznamu. Tento proces se děje do doby, než bude dodán kompletní seznam.

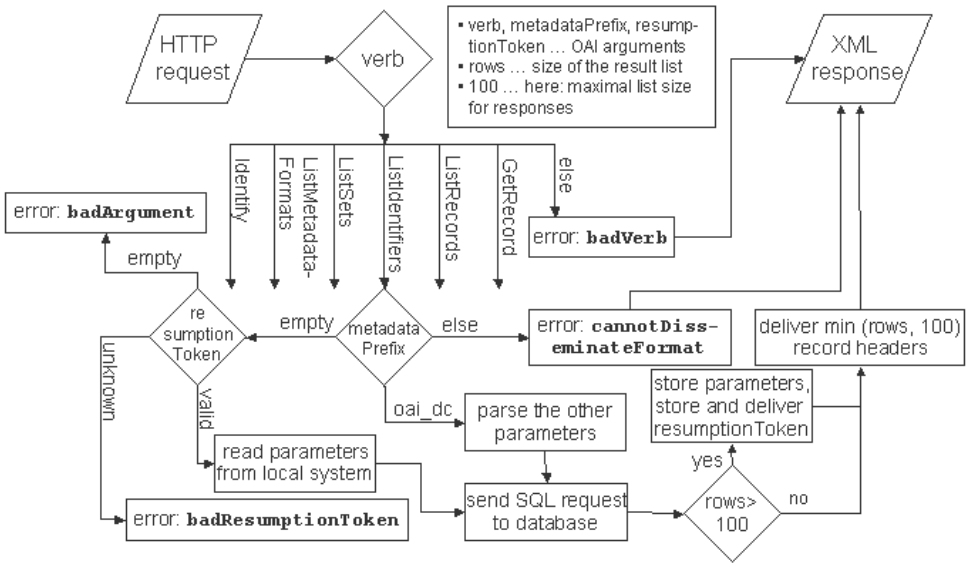
Obrázek 2.4 vyobrazuje ukázku zpracování dotazu protokolu OAI-PMH, zde je příklad uveden na získávání seznamu identifikátorů pomocí klíčového slova *ListIdentifiers* i s možnými chybovými stavy a rozdělením seznamu na více částí.

Po příchodu dotazu na server, se podle zjištěného klíčového slova zvolí větev, která dotaz zpracuje. V našem případě se jedná o větev, která zpracovává dotazy s klíčovým slovem *ListIdentifiers*. Následně se přečte atribut *metadataPrefix*. V případě, že se zde nachází hodnota *oai_dc* (požadovaný metadatový

[4]XSD soubor je dostupný na webových stránkách organizace OAI na adrese: `http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OAI-PMH xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:m="http://www.loc.gov/MARC21/slim"
xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns="http://www.openarchives.org/OAI/2.0/">
    <responseDate>2019-04-18T07:46:18.539Z</responseDate>
    <request until="2019-04-18" from="1970-01-01"
        verb="Identify">https://biblio.hiu.cas.cz/api/oai</request>
  - <Identify>
        <repositoryName>Historický ústav AV</repositoryName>
        <baseURL>https://biblio.hiu.cas.cz/api/oai</baseURL>
        <protocolVersion>2.0</protocolVersion>
        <adminEmail>horcakova@hiu.cas.cz</adminEmail>
        <earliestDatestamp>1900-01-01</earliestDatestamp>
        <deletedRecord>persistent</deletedRecord>
        <granularity>YYYY-MM-DD</granularity>
        <compression>no</compression>
    </Identify>
</OAI-PMH>
```

Obrázek 2.3: OAI-PMH - příklad odpovědi na dotaz *GetRecord*



Obrázek 2.4: Ukázka zpracování dotazu protokolu OAI-PMH [18]

formát je Dublin Core), zpracují se i další parametry a položí se dotaz do databáze. Pokud je v tomto atributu specifikován formát, který server neumí zpracovávat, vrátí klientovi chybu typu *cannotDisseminateFormat*. V případě, že hodnota není nastavena, přechází se na kontrolu atributu *resumptionToken*.

Jestliže atribut *resumptionToken* není vyplněn, server odpovídá klientovi chybou typu *BadArgument*, v opačném případě dojde k validaci vlastního tokenu. Pokud je token znám, v lokální paměti vyhledá potřebné informace podle získaného tokenu a pošle dotaz do databáze. Pokud token není serveru znám, vrátí klientovi chybu *badResumptionToken*. Po získání záznamů z databáze se podle jejich počtu rozhodne, zda se k odpovědi přidá *resumptionToken*. Pokud je záznamů více než sto, vygeneruje se *resumptionToken*, který se připojí k odesílané odpovědi spolu se záznamy, které se omezí pouze na počet 100. Do lokální paměti se uloží *resumptionToken* a další informace potřebné k tomu, aby při jeho použití v dalším dotazu byl server schopný dodat další část seznamu identifikátorů.

V následují tabulce 2.1 jsou ve stručnosti shrnuty výhody a nevýhody protokolu OAI-PMH vzhledem k použití ve Vokabuláři webovém.

Výhody	**Nevýhody**
Rozšířenost	Přílišná obecnost
Menší zatížení sítě díky inkrementálnímu sklízení záznamů	Neaktuálnost dat při modifikaci v repozitáři
Relativně snadná implementace jak na straně klienta, tak na straně serveru	
Libovolné XML schéma pro data	

Tabulka 2.1: Výhody a nevýhody protokolu OAI-PMH

2.5.2 Protokol Z39.50

Z39.50 je standardizovaný vyhledávací protokol, který specifikuje komunikaci mezi serverem a klientem pro vyhledávání informací z databází, ze kterých server agreguje data [19].

Historie protokolu

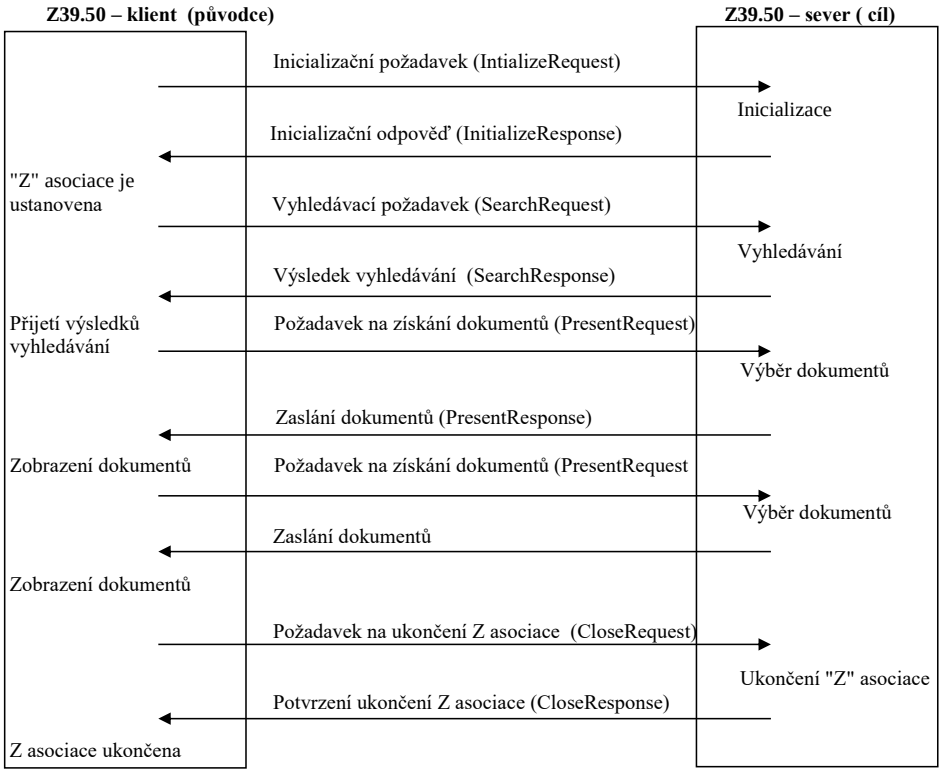
Vývoj protokolu začal v 70. letech minulého století za účelem sdílet katalogizaci mezi knihovnami. První verze protokolu byla vydána až v roce 1988 s označením Z39.50-1988. Roku 1997 byla verze 3 uznána jako ISO standard s označením ISO 23950 [20].

Využití protokolu

Protokol našel využití zejména v knihovnictví. Využíván je zde k dálkovému prohledávání. Kromě toho je sjednocujícím prvkem při sdílení dat mezi různými knihovnickými systémy [20].

Podstata protokolu

Aplikační protokol Z39.50 funguje na komunikaci mezi klientem a serverem. Protokol je stavový, tudíž vyžaduje dodržení posloupnosti volaných funkcí v rámci relace neboli v rámci „Z“ asociace [19]. Průběh komunikace je znázorněn na obrázku 2.5. Komunikace probíhá tak, že klient pošle serveru inicializační požadavek, na základě hodnot parametrů se vytvoření relace a server odpoví klientovi s inicializační odpovědí, ve které souhlasí vytvořením „Z“ asociace. Následně klient odešle vyhledávací požadavek, server posílá zpět po prohledání databáze v odpovědi výsledek vyhledávání. Pro ukončení „Z“ asociace dochází k požadavku na ukončení, který může učinit klient i server, následuje potvrzení u ukončení „Z“ asociace [21].



Obrázek 2.5: Příklad komunikace pomocí protokolu Z39.50 [19]

Struktura protokolu

Protokol má definovaných celkem 11 funkcí. Většina funkcí je složena ze skupiny služeb. Služby jsou dvojího typu:

- Potvrzované – vyžadují zpětnou reakci volaného.
- Nepotvrzované – nevyžadují zpětnou reakci volaného.

Funkce:

- Inicializační funkce.
- Vyhledávací funkce.
- Funkce pro získání dat.
  - Služba pro získávání dat.
  - Služba pro segmentaci.
- Funkce vymazání množiny výsledků: potvrzovaná.
- Funkce pro prohlížení.
- Funkce pro řazení.
- Funkce pro řízení přístupu.
- Funkce pro řízení účtování/zdroje.
- Funkce vysvětlení.
- Funkce rozšířených služeb.
- Funkce ukončení.

Při pokládání vyhledávacích dotazů se využívají výrazy a atributy. Výraz je hodnota, kterou chceme nalézt. Atributy specifikují, co hledaný výraz musí splňovat, například určíme, že hledaný výraz je typu autor. Každý atribut spadá do určité množiny atributů. Tyto množiny mají své identifikátory, například v knihovnictví se používá množina atributů s identifikátorem Bib-1. Tyto množiny slouží ke specifikování seznamu atributů, ve kterém lze nalézt jejich významy [19].

Celková složitost protokolu Z39.50 a množství atributů vedlo k vytvoření profilů, které specifikují, které funkce a atributy daný Z-server podporuje. Nejrozšířenějším profilem je mezinárodně uznávaný profil Bath [19].

V tabulce 2.2 jsou stručně shrnuty výhody a nevýhody protokolu Z39.50 vzhledem k použití ve Vokabuláři webovém.

Výhody	Nevýhody
Možnost filtrování	Neaktuálnost dat při modifikaci v repozitáři
Možnost vyhledávání	Podpora pouze u jednoho z dodaných zdrojů

Tabulka 2.2: Výhody a nevýhody protokolu Z39.50

2.5.3 Porovnání protokolů

Protokol Z39.50, poskytuje API pro vyhledávání. Oproti tomu protokol OAI-PMH poskytuje API pro periodické získávání záznamů, které byly za určitý časový úsek změněny. Značnou nevýhodou protokolu Z39.50 při použití ve Vokabuláři webovém je využití pouze u jednoho z dodaných zdrojů. Tudíž pro použití ve Vokabuláři webovém dle definovaných funkčních a nefunkčních požadavků se jeví více vhodný protokol OAI-PMH.

2.6 Analýza standardů pro popis metadat

Tato sekce popisuje standardy, jenž udávají způsoby a pravidla pro popis metadat. Prvním popsaným standardem je Dublin Core, jenž je jednoduchý a rozšířený mezi laickou veřejností. Druhým standardem je MARC 21, který je dosti robustní a složitý, využívá se převážně v knihovnictví.

2.6.1 Dublin Core

Dublin Core je jeden z nejjednodušších standardů pro metadatový popis digitálních informací. Dublin Core je nejčastěji vyjadřován pomocí značkovacího jazyka XML [22]. Standard je spravován a rozvíjen organizací Dublin Core Metadata Initiative (dále jen DCMI). Vytvořen byl v roce 1995. Cílem bylo vytvořit metadatový formát, který bude sloužit k jednoduchému popisu webových objektů, ale díky jeho jednoduchosti začalo docházet k využití pro popis jiných druhů dokumentů – například knih a hudby, potažmo všeho, co vlastnostmi odpovídá textovému dokumentu [22].

Úrovně standardu

Standard je ve verzi 1.1 tvořen souborem 15 základních prvků, z nichž ani jeden není povinný a jednotlivé prvky se mohou opakovat v libovolném

počtu a pořadí. Fungují na principu klíč–hodnota. Na obrázku 2.6 jsou metadata ve standardu Dublin Core ve formátu XML získaných skrze OAI-PMH. Na obrázku lze vidět využití několika základních prvků – title, creator, subject a language. Tento soubor je též označován jako nekvalifikovaný Dublin Core.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OAI-PMH xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:m="http://www.loc.gov/MARC21/slim"
xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns="http://www.openarchives.org/OAI/2.0/">
    <responseDate>2019-04-17T21:10:55.277Z</responseDate>
    <request until="2019-04-17" from="1970-01-01" metadataPrefix="oai_dc"
        identifier="oai:biblio.hiu.cas.cz:361543"
        verb="GetRecord">https://biblio.hiu.cas.cz/api/oai</request>
  - <GetRecord>
      - <record>
          - <header>
                <identifier>oai:biblio.hiu.cas.cz:361543</identifier>
            </header>
          - <metadata>
              - <oai_dc:dc>
                    <dc:title>Knihy kacířů se mají číst</dc:title>
                    <dc:creator>Hus, Jan</dc:creator>
                    <dc:subject>myšlení náboženské</dc:subject>
                    <dc:subject>reformátoři</dc:subject>
                    <dc:subject>teologie křesťanská</dc:subject>
                    <dc:subject>edice</dc:subject>
                    <dc:subject>studie</dc:subject>
                    <dc:language>cze</dc:language>
                </oai_dc:dc>
            </metadata>
        </record>
    </GetRecord>
</OAI-PMH>
```

Obrázek 2.6: Metadata ve standardu Dublin Core

Z důvodu větší flexibility standardu byl vytvořen další formát – kvalifikovaný Dublin Core je nadstavbou nad nekvalifikovaným Dublin Core formátem. Kvalifikovaný DC obsahuje kvalifikátory, jež specifikují základní prvky. Existují kvalifikátory dvojího typu [23]:

- Kvalifikátory upřesňující význam prvku – tyto kvalifikátory činí význam prvku více specifičtější. Například pokud máme prvek Date, můžeme k němu přidat kvalifikátor Created, který značí, že se jedná o datum vytvoření metadat.

- Kvalifikátory upřesňující schéma zápisu – tyto kvalifikátory určují, jaké schéma musí splňovat hodnota v daném prvku, který tento kvalifikátor zpřesňuje. Například pro prvek Date můžeme určit, v jakém pořadí budou informace o datu – rok, měsíc, den.

Kvalifikovaný Dublin Core byl nahrazen specifikací DCMI Metadata Terms [24]. Avšak s termínem kvalifikovaný Dublin Core se můžeme stále setkat.

DCMI Metadata Terms neboli termíny metadat DCMI je směrodatná specifikace všech metadatových termínů spravovaných organizací DCMI. Metadatové termíny obsahují prvky, zpřesnění prvků, schémata zápisů a slovníky termínů [25].

2.6.2 MARC 21

MARC (Machine Readable Cataloging) je označení pro skupinu standardizovaných strojově čitelných formátů, které určují strukturu katalogizačních (popřípadě bibliografických) záznamů. Slouží zejména pro výměnu dat mezi jednotlivými informačními systémy [26]. Standardy MARC určují popis obsahu – kódy a znaky a jejich význam k obsahu samotnému. Struktura tohoto popisu vychází z mezinárodního komunikačního standardu ISO 2709 známého také jako ANSI/NISO Z39.2 [27].

Standard MARC 21 je výsledkem kombinace standardů USMARC (United States MARC) a CAN/MARC (Canadian MARC). Vznikl v roce 1999 za účelem standardizace výměnných knihovnických formátů na mezinárodní úrovni. Na jeho vývoji se podílely knihovny Library of Congress, Library and Archives Canada, British Library. V současné době se o jeho dokumentaci stará Library of Congress. Neustálý vývoj standardu MARC 21 umožňuje zaznamenávání maxima údajů o různých typech dokumentu [26]. O rozvoj formátu se stará MARC Steering Group, jenž je složena z výše vyjmenovaných knihoven a několika dalších národních knihoven jiných zemí [28].

Po uvedení začal nahrazovat své předchůdce a jim podobné formáty, například v Evropě nahradil mezinárodní formát UNIMARC (Universal MARC). Oproti standardu UNIMARC má lepší finanční i personální podporu, rychleji reaguje na nové požadavky a v neposlední řadě vede k jednoduché výměně mezi institucemi díky standardizaci [29].

Využití MARC21

Formát MARC 21 se používá především v knihovnické komunitě, jelikož metadata vycházející ze standardu MARC 21 jsou velmi složitá, což stěžuje jejich tvorbu a další práci s nimi. Provést katalogizaci uměleckého díla ve formátu

MARC 21 je velmi komplikovaný proces, který zvládne pouze profesionální katalogizátor. Proto není příliš rozšířen mezi veřejností.

Struktura standardu MARC 21

Standard MARC 21 obsahuje formáty pro následující typy dat [27]:

- bibliografická data,
- autoritní data,
- data o vlastnících dokumentů,
- klasifikační data,
- data o společnostech.

Ve Vokabuláři webovém bude využit formát pro bibliografická data, jenž nese informace o dílech. Záznam ve standardu MARC 21 můžeme rozdělit na následující části:

- Návěští.
- Proměnná pole.
  - Proměnná kontrolní pole.
  - Proměnná pole údajů.

Návěští

Návěští je první pole každého MARC záznamu, obsahuje základní údaje potřebné pro zpracování záznamu. Tyto jsou určené pro strojové zpracování, pole obsahuje kódované údaje – znaky na jednotlivých pozicích mají přesné definice významů. Návěští celkově obsahuje 24 znaků [30].

Proměnná pole

Proměnná pole obsahují již vlastní údaje ohledně popisovaného díla. Každé pole má svoje označení neboli tag [30]. Proměnná pole lze dále dělit na proměnná kontrolní pole a proměnná pole údajů.

Proměnná kontrolní pole

Proměnná kontrolní pole obsahují základní údaje, které se využívají pro zpracování záznamů. Označení kontrolních polí začíná vždy dvěma nulami.

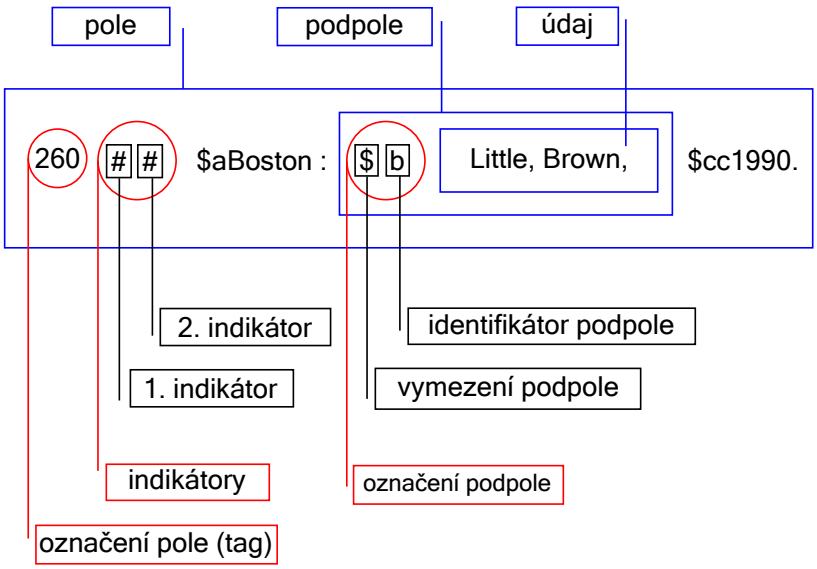
Tudíž proměnná kontrolní pole jsou pole s tagy 001–009. U kontrolních polí rozlišujeme jejich dva typy. Prvním typem jsou pole, která obsahují pouze jeden údaj. Druhým typem polí jsou pole kódovaná, jejich obsah se skládá z několika kódovaných údajů obdobně jako u návěští [30].

Proměnná pole údajů

Proměnná pole údajů obsahují bibliografické údaje o daném díle. Tvoří je všechna proměnná pole (01X–9XX) kromě proměnných kontrolních polí. Proměnná pole údajů můžou být opakovatelná a neopakovatelná. Tyto pole se skládají z následujících údajů [30]:

- Označení pole (tag).
- Dva indikátory.
- Označení podpolí.
- Údaj.

Grafické znázornění struktury proměnného pole údajů lze vidět na obrázku 2.7, v němž je využit tzv. řádkový zápis formátu MARC 21. Zobrazené pole s tagem 260 obsahuje nakladatelské údaje [31]. Podpole s identifikátorem *a* je místo vydání díla, identifikátor *b* specifikuje název nakladatelství a poslední identifikátor *c* značí, v jakém roce bylo dílo vydáno.



Obrázek 2.7: Struktura MARC 21 proměnného pole údajů (řádkový zápis) [32]

Indikátory

Indikátory jsou doplňující prvky, které vysvětlují nebo doplňují informace o údaji obsaženém v poli. Pro každé pole jsou definovány zvlášť, nezávisle na sobě. Každé pole může mít definovány celkově až 2 indikátory, ale nemusejí mít ani jeden. Indikátory jsou reprezentovány abecedním znakem, číslem nebo mezerou (ta zpravidla značí, že indikátor není definován pro toto pole). Hodnoty indikátorů jsou interpretovány nezávisle na sobě [30].

Podpole

Podpole dále strukturuje údaj uvnitř pole. Jednotlivá podpole mají svá identifikátory, jedná se buď o abecední nebo číselný znak. Identifikátory podpolí jsou definovány nezávisle, pro každé pole zvlášť. Každé pole obsahuje minimálně jedno označené podpole [30].

Proměnné pole údajů jsou děleny do devíti bloků podle prvního znaku v identifikátoru. Názvy bloků se nachází v následujícím seznamu [27]:

- 0XX – Kontrolní informace, identifikační čísla, kontrolní pole, klasifikační znaky atd.
- 1XX – Hlavní záhlaví.
- 2XX – Názvy a údaje o vydání, nakladatelské údaje atd.
- 3XX – Údaje fyzického popisu atd.
- 4XX – Údaje o edici.
- 5XX – Poznámky.
- 6XX – Věcné selekční údaje.
- 7XX – Vedlejší záhlaví s výjimkou předmětů a edic; propojovací pole.
- 8XX – Vedlejší záhlaví pro edici; knihovní jednotky atd.
- 9XX – Pole rezervována pro národní použití.

Příklad metadata ve standardu MARC 21

V následujícím obrázku 2.8 jsou zobrazena metadata díla O smyslu českých dějin. Metadata byla získána z webových stránek Historického ústavu AV ČR[5]. Metadata jsou zapsána v řádkovém formátu standardu MARC 21.

[5]Metadata jsou dostupná z: `https://biblio.hiu.cas.cz/documents/1`.

První pole je již zmíněné návěští. Následují tři kontrolní pole. Kontrolní pole s označením 001 (kontrolní číslo) a 003 (identifikátor kontrolního čísla) jsou pole s jedním údajem. Pole s označením 008 (kontrolní pole) je pole s kódovanými údaji [31]. Dále jsou již pouze proměnná pole údajů, u nich přibývají indikátory, v tomto případě nedefinované identifikátory jsou značeny znakem # místo mezery. V poli označeném číslem 100 (osobní jméno – hlavní záhlaví) se nachází údaje o autorovi. První indikátor s hodnotou 1 značí, v podpoli *a* se nachází jako první údaj příjmení autora. Dále stojí za zmínku pole 245 (údaje o názvu). Tyto dvě uvedená proměnná pole údajů spadají do kategorie neopakovatelných polí na rozdíl od polí 653 (klíčové slovo), 655 (žánr/forma), 659 (systematika), která jsou opakovatelná.

```
001       kpm011
003       CZ-PrHAC
008       120730s1990----cze||||f||||||||||xr|||
072   #7  $a 94(437) $x Dějiny Česka a Slovenska $2 Konspekt $9 8
100   1#  $a Pekař, Josef, $d 1870-1937 $7 jk01092389 $4 aut
245   10  $a O smyslu českých dějin / $c Josef Pekař
260   ##  $a Praha : $b Rozmluvy, $c 1990
300   ##  $a 418 s.
653   0#  $a historiografie česká
653   0#  $a filozofie dějin
655   #7  $a studie
655   #7  $a soubory studií
659   ##  $a xab $x filozofie dějin
659   ##  $a yba $x přehledná zpracování dějin českých zemí (chronologicky)
659   ##  $a zza $x dějepisectví, historické vědy, historici
```

Obrázek 2.8: Příklad metadat ve standardu MARC 21 v řádkovém zápisu

MARCXML

MARCXML je metadatové XML schéma, jenž se řídí standardem MARC 21. Toto schéma je definované pomocí XSD souboru[6] [33].

Vznik

Důvodem vzniku metadatového schématu MARCXML byla snaha o modernizaci a rozšíření standardu MARC. Jelikož pro práci s formátem MARC 21 existuje poměrně malé množství nástrojů, uvedením MARCXML se možnosti značně rozšiřují, jelikož nástrojů podporujících XML je značné množství, z nichž velká část je volně dostupná [34].

Vlastnosti MARCXML se dají shrnout do několika základních bodů [34]:

- Rámec pro práci s daty ve standardu MARC v XML prostředí.
- Rámec má podobu metadatového XML schématu, které obsahuje MARC data.
- Umožňuje bezeztrátovou konverzi mezi následujícími formáty:
  - Z MARC do MARCXML.
  - Z MARCXML do MARC.
  - Z MARCXML do jiných formátů, které jsou založeny na XML.
- Je určen pro výměnu a sdílení záznamů.
- Poskytuje validaci dat, kterou lze rozdělit do tří úrovní:
  - Základní XML validace vzhledem ke schématu MARCXML.
  - Validace MARC 21 označení polí a podpolí.
  - Validace obsahu - kódované hodnoty, data, časy.

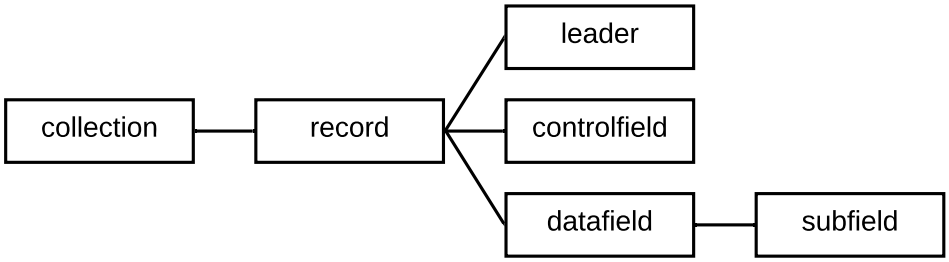
Struktura

Jelikož formát MARCXML je založen na standardu MARC 21, kopíruje schéma jeho strukturu. Celkem struktura obsahuje šest základních elementů – *collection, record, leader, controlfield, datafield, subfield* (viz obrázek 2.9).

Collection

Element *collection* je kořenový element, jedná se o množinu záznamů. Obsahuje v sobě sekvenci záznamů neboli elementů typu *record*. Tato množina záznamů může být i prázdná [34].

[6] XSD soubor je dostupný z: `http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd`.



Obrázek 2.9: Elementy MARCXML [34]

Record

Record neboli záznam je element, který může nahradit element *collection* a sám být kořenovým elementem. Záznam je složen ze tří typů elementů: *leader* (návěští), *controlfield* (proměnné kontrolní pole), *datafield* (proměnné pole údajů). Element *record* může obsahovat více elementů typu *controlfield* a *datafield* [34].

Element *record* může obsahovat i atribut *type*, který je výčtem následujících hodnot:

- *bibliographic*,
- *authority*,
- *holdings*,
- *classification*,
- *community*.

Tento výčet odpovídá typům formátů standardu MARC 21. Tudíž pro tuto práci jsou relevantní záznamy typu *bibliographic*.

Leader

Element *leader* odpovídá poli návěští ve standardu MARC 21. Obdobně jako pole návěští má i element *leader* omezen délku na 24 znaků, kromě toho má definovanou masku pomocí regulárních výrazů, kterou musí splňovat. Ve struktuře dokumentu se může nacházet pouze jednou [34].

Controlfield

Controlfield – proměnné kontrolní pole. Obsahuje povinný atribut *tag*, který odpovídá označení pole v MARC 21, tudíž může nabývat pouze hodnot 001–009. V dokumentu může být obsaženo v libovolném počtu [34].

Datafield

Datafield – proměnné pole údajů, poslední přímý potomek elementu *record*. Obsahuje sekvencí podpolí – elementy *subfield*. Má definované následující tři atributy:

- *tag* – označení pole,
- *ind1* – indikátor 1,
- *ind2* — indikátor 2.

Subfield

Subfield (podpole) jsou potomci elementu *datafield*. Má povinný atribut *code*, jenž odpovídá identifikátoru podpole dle standardu MARC 21. Obsahem podpole jsou již řetězce znaků – vlastní údaje [34].

Atribut ID

Výše zmíněné elementy mají nepovinný atribut *ID*, jenž musí mí unikátní hodnotu v rámci elementu nebo komplexního typu [34].

Příklad metadat ve formátu MARCXML

Na obrázku 2.10 jsou zobrazena metadata díla O smyslu českých dějin. Obdobně jako metadata na obrázku 2.8 byla tato metadata získána z webových stránek Historického ústavu AV ČR. Rozdíl je pouze v tom, že byla získána skrze protokol OAI-PMH[7]. Metadata jsou zapsána v formátu MARCXML.

Tento příklad obsahuje stejná pole jako příklad pro standard MARC 21. Indikátory, které nejsou definovány jsou zde značeny mezerou.

[7]Metadata jsou dostupná z: `https://biblio.hiu.cas.cz/api/oai?verb=GetRecord&metadataPrefix=marc21&identifier=oai:biblio.hiu.cas.cz:1`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OAI-PMH xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:m="http://www.loc.gov/MARC21/slim"
xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
xmlns="http://www.openarchives.org/OAI/2.0/">
    <responseDate>2019-04-18T07:30:17.099Z</responseDate>
    <request until="2019-04-18" from="1970-01-01" metadataPrefix="marc21"
        identifier="oai:biblio.hiu.cas.cz:361543"
        verb="GetRecord">https://biblio.hiu.cas.cz/api/oai</request>
  - <GetRecord>
      - <record>
          - <header>
              <identifier>oai:biblio.hiu.cas.cz:361543</identifier>
            </header>
          - <metadata>
              - <m:record>
                  <m:leader>-----nam-a22------i-4500</m:leader>
                  <m:controlfield tag="001">kpm01361543</m:controlfield>
                  <m:controlfield tag="003">CZ-PrHAC</m:controlfield>
                  <m:controlfield tag="008">151126s2015----xr ||||e|||||001|0dcze||</m:controlfield>
                + <m:datafield tag="020" ind2=" " ind1=" ">
                + <m:datafield tag="040" ind2=" " ind1=" ">
                + <m:datafield tag="072" ind2="7" ind1=" ">
                + <m:datafield tag="080" ind2=" " ind1=" ">
                - <m:datafield tag="100" ind2=" " ind1="1">
                      <m:subfield code="a">Hus, Jan,</m:subfield>
                      <m:subfield code="d">asi 1371-1415</m:subfield>
                      <m:subfield code="e">Autor</m:subfield>
                  </m:datafield>
                - <m:datafield tag="245" ind2="0" ind1="1">
                      <m:subfield code="a">Knihy kacířů se mají číst /</m:subfield>
                      <m:subfield code="c">Mistr Jan Hus ; z latinského originálu poprvé přeložil a
                          doprovodným slovem opatřil Martin Wernisch</m:subfield>
                  </m:datafield>
                + <m:datafield tag="250" ind2=" " ind1=" ">
                + <m:datafield tag="264" ind2="1" ind1=" ">
                + <m:datafield tag="653" ind2=" " ind1=" ">
                + <m:datafield tag="653" ind2=" " ind1=" ">
                - <m:datafield tag="653" ind2=" " ind1=" ">
                      <m:subfield code="a">teologie křesťanská</m:subfield>
                  </m:datafield>
                - <m:datafield tag="655" ind2="7" ind1=" ">
                      <m:subfield code="a">edice</m:subfield>
                  </m:datafield>
                + <m:datafield tag="655" ind2="7" ind1=" ">
                + <m:datafield tag="659" ind2=" " ind1=" ">
              </m:record>
          </metadata>
      </record>
  </GetRecord>
</OAI-PMH>
```

Obrázek 2.10: MARCXML – příklad

2.6.3 Porovnání metadatových formátů

Pro účely Vokabuláře webového je lepší získávat data z externích bibliografický zdrojů ve formátu MARC 21, jelikož je oproti formátu Dublin Core robustnější a obsahuje lepší identifikaci polí, proto lze jednotlivá pole navázat na entity ve Vokabuláři webovém. U formátu Dublin Core tohle není možné realizovat, jelikož pole nemají jednoznačné identifikátory.

2.7 Analýza dostupných knihoven

Na základě předchozích kapitol bude zvolen protokol OAI-PMH pro implementaci ve Vokabuláři webovém. Tudíž byly zváženy knihovny implementující OAI-PMH data harvester, tedy klientské aplikace získávající data pomocí OAI-PMH protokolu, které by vyhovovaly pro použití ve Vokabuláři webovém.

Při výběru knihovny je nejdůležitější, aby šla integrovat do Vokabuláře webového. Tudíž musí využívat platformu .NET Core nebo splňovat rozhraní .NET Standard. Knihovna musí být vydána pod open source licencí kompatibilní s licencí Vokabuláře webového (Vokabulář webový je vydán pod BSD licencí[8]). Dále by kód knihovny měl být přehledný a udržovaný. Následující hodnocení knihoven je tudíž vztaženo vzhledem k použití ve Vokabuláři webovém.

2.7.1 Zvažované knihovny

Následuje seznam zvažovaných knihoven pro použití ve Vokabuláři webovém, jako klienta, který bude získávat data pomocí protokolu OAI-PMH:

Oai.cs

Vcelku jednoduchá knihovna pro práci s rozhraním protokolu OAI-PMH, psána je v jazyce C#. Knihovnu vyvinul v roce 2004 Terry Resse z The Ohio State University. Od té doby neprošla žádnými většími změnami. Kód není příliš čitelný a strukturovaný, využívá technologie, které jsou již zastaralé. Stavěn je na platformě .NET Framework, verze 4.6.1, což je největší nevýhoda vzhledem k požadavku na kompatibilitu technologií s Vokabulářem webovým, který využívá možností platforma ASP.NET Core verze 2.1.

Knihovna je vydána pod open source licencí GNU General Public License[9]. Tato licence není kompatibilní s licencí BSD. Dostupná je na serveru GitHub na adrese `https://github.com/reeset/oai.cs`. V následují tabulce 2.3 jsou shrnuty výhody a nevýhody knihovny Oai.cs.

Výhody	Nevýhody
Jednoduchost	Nepřehledný kód
	.NET Framework 4.6.1
	GPL licence
	Zastaralé technologie

Tabulka 2.3: Výhody a nevýhody knihovny Oai.cs

[8] Licence je dostupná z: `https://github.com/RIDICS/ITJakub/blob/master/LICENSE`.
[9] Znění licence je dostupné z: `http://www.gnu.org/licenses/gpl.txt`.

OAI-PMH-.Net

OAI-PMH-.Net je MVC aplikace jenž poskytuje, jak sběrač dat – klientská strana, tak i poskytovatele dat – serverová strana. Aplikace jako celek nesplňuje požadavky pro přímou integraci do portálu Vokabuláře webového. Aplikace již není aktivně vyvíjena[10]. Použití aplikace jako knihovny pro komunikaci pomocí protokolu OAI-PMH se zde v tomto případě nehodí, jelikož komunikace je úzce provázána s přístupem do databáze, který je realizován pomocí knihovny Entity Framework, který se ve Vokabuláři webovém nevyužívá, kromě toho se zde nachází poměrně robustní nevyužitá část, která realizuje OAI-PMH data provider – serverovou část.

Aplikace je vydána pod open source licencí GNU General Public License. Dostupná je na serveru GitHub na adrese `https://github.com/kkrajnc/OAI-PMH-.Net`. V následující tabulce 2.4 jsou shrnuty výhody a nevýhody knihovny OAI-PMH-.Net.

Výhody	Nevýhody
Podpora uložení konfigurace zdroje	.NET Framework 4.5
	GPL licence
	Robustní MVC aplikace

Tabulka 2.4: Výhody a nevýhody knihovny OAI-PMH-.Net

OaiHarvestAndStore

Knihovna OaiHarvestAndStore poskytuje repozitář pro poskytování dat i klienta pro sklízení dat. Pro komunikaci pomocí protokolu OAI-PMH používá již zmíněnou knihovnu Oai.cs. Jelikož je potřeba knihovny pro komunikaci, bylo by v tomto případě efektivnější použít přímo knihovnu Oai.cs, bez dalších nadbytečných úprav. Kód je velmi nepřehledný, neudržovaný a jsou využity zastaralé technologie.

Knihovna je dostupná na serveru GitHub: `https://github.com/rodricios/OaiHarvestAndStore`. Licence není uvedena.

Výhody	Nevýhody
	.NET Framework 4.5
	Neuvedena licence
	Nepřehledný kód

Tabulka 2.5: Výhody a nevýhody knihovny OaiHarvestAndStore

[10]Poslední commit v repozitáři byl v roce 2014, repozitář je dostupný z `https://github.com/kkrajnc/OAI-PMH-.Net`.

OaiPmhNet

Tato knihovna je sice kompatibilní s Vokabulářem webovým – využívá .NET Standard 2.0, ale neposkytuje potřebnou funkcionalitu, tj. klienta pro získávání dat z repozitáře. V knihovně je implementován pouze repozitář, jenž poskytuje data pomocí protokolu OAI-PMH.

Knihovna je dostupná z: `https://github.com/niklr/OaiPmhNet`

Ostatní knihovny

Kromě knihoven psaných v programovacím jazyce C# je dostupné množství knihoven, jenž souvisí s komunikací skrze OAI psaných v jiných programovacích jazycích. Tyto knihovny jsou psány nejčastěji v programovacích jazycích Java, Python, PHP[11]. Tyto knihovny nebyly podrobeny podrobnější analýze z důvodu toho, že je nelze integrovat do Vokabuláře webového, jelikož používají odlišné technologie.

2.7.2 Porovnání knihoven

Žádná z analyzovaných knihoven nesplňuje požadavky, zejména na kompatibilitu s technologiemi Vokabuláře webového, často by muselo dojít i k modifikaci kódu, aby daná knihovna byla vhodná pro použití. Nejlepším řešením je implementovat vlastní knihovnu pro komunikaci pomocí protokolu OAI-PMH, jelikož protokol není nijak výrazně složitý a tudíž i klient nebude obtížný k implementování.

[11] Bráno dle nalezených repozitářů na serveru GitHub, dostupné z `https://github.com/search?q=oai`.

Kapitola 3

Návrh

Návrh řešení se zabývá záležitostmi ohledně celkové architektury služeb a komunikace mezi Vokabulářem webovým a externími bibliografickými knihovnami pomocí analyzovaného protokolu OAI-PMH. Dále je zde uveden návrh architektury pro konvertor dat z formátu MARC 21 do entit Vokabuláře webového. V poslední části kapitoly se nachází UI návrhy pro nastavení a zobrazení detailů externích bibliografických databází, ze kterých se budou metadata získávat.

3.1 Architektura služby

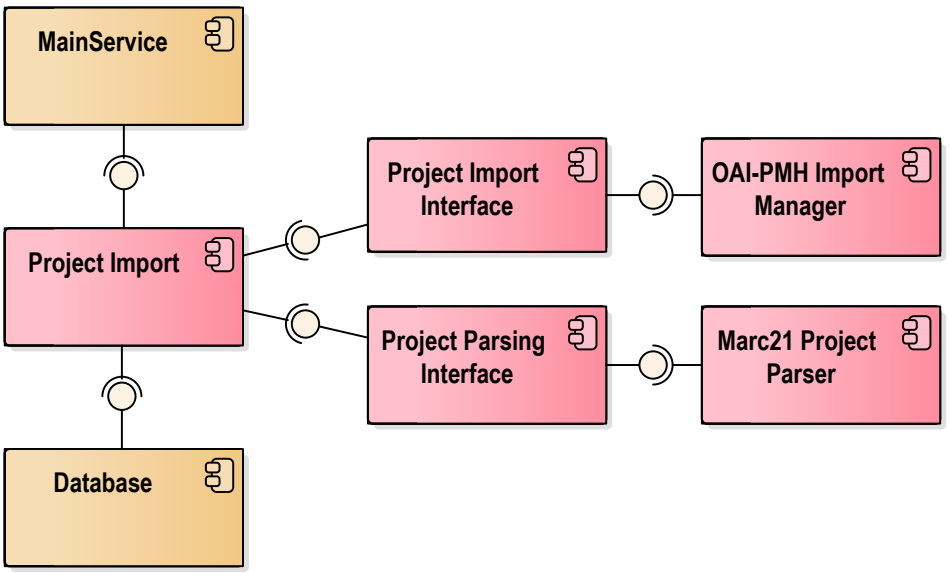
Jednotlivé služby pro komunikaci a zpracování XML metadat ve standardu MARC21 budou navrženy jako samostatné knihovny. S těmito knihovnami bude operovat importní knihovna, která bude zajišťovat uložení dat ve Vokabuláři webovém, a to jak získaných metadat, tak i například konfigurace externích bibliografických databází.

Všechny služby budou využívat návrhového vzoru Inversion of Control, stejně jako celý projekt Vokabulář webový [7]. Pro přístup k relační databázi bude vhodné použít ORM framework NHibernate, jenž je použitý ve zbytku Vokabuláře webového.

Modul pro importování děl z externích bibliografických zdrojů bude využíván hlavní službou Vokabuláře webového (viz obrázek 3.1). Vstupním bodem bude knihovna pro importování děl neboli projektů. Tato knihovna bude mít přístup k relační databázi Vokabuláře webového, kam bude ukládat metadata děl získaných z externích bibliografických databází.

Spojení s příslušným externím bibliografickým zdrojem jí poskytne daná komunikační knihovna, která bude implementovat rozhraní definované sdílenou knihovnou mezi knihovnami pro import a komunikaci. Převod získaných

dat na entity Vokabuláře webového se provede pomocí příslušné knihovny pro konverzi dat, která bude obdobně jako u komunikační knihovny splňovat dané rozhraní definované sdílenou knihovnou mezi těmito knihovnami.



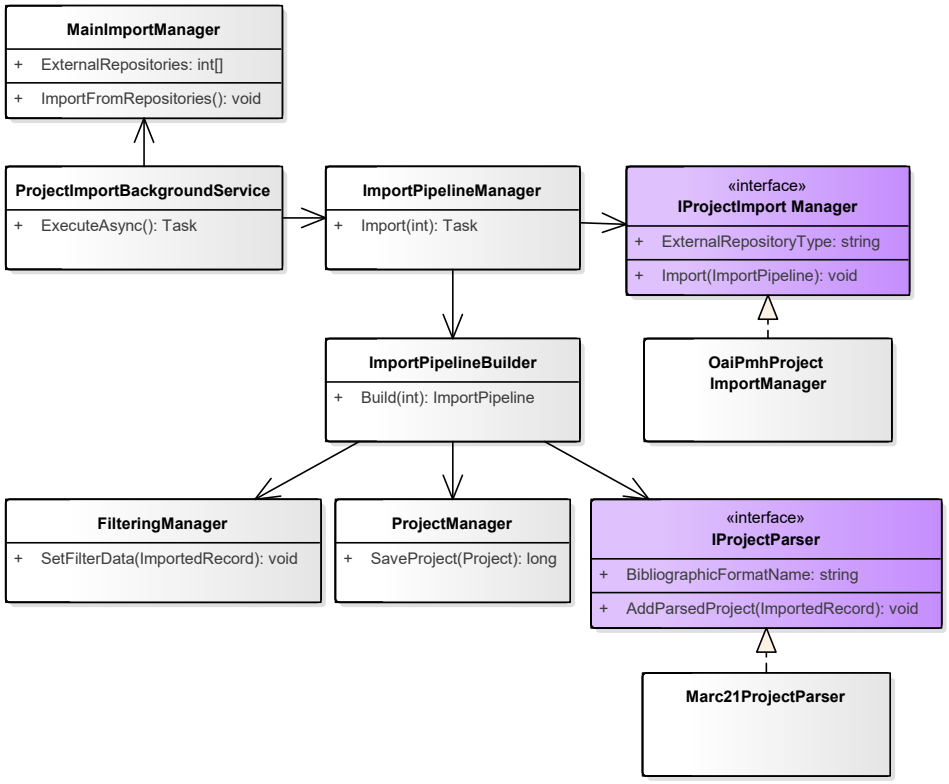
Obrázek 3.1: Model komponent

3.1.1 Návrh procesu importu

Získávání dat z externích bibliografických databází (dále jen zdroje) bude rozděleno do několika knihoven. Vstupním bodem celého procesu importu děl bude knihovna pro importování projektů.

V knihovně pro import projektů se bude nacházet hlavní importní manažer, který bude mít metodu přijímací seznam zdrojů. Tyto zdroje mu dodá hlavní importní služba při zahájení procesu importu (viz obrázek 3.2). Dále se zde bude nacházet třída, jenž bude zajišťovat běh importu. Samotný import potrvá dále, než HTTP požadavek, který ho vyvolal [35]. Proto je nutné, aby tato třída běžela v samostatné službě, která nebude závislá na čase určeném pro zpracování HTTP požadavku.

V rámci ASP.NET se nabízí použití třídy *BackgroundService* z knihovny Microsoft.Extension.Hosting.BackgroundService [36]. Tato třída poskytuje podporu pro běh třídy na pozadí aplikace jako samostatné služby. Výhodou oproti vytváření samostatné REST služby je snadná a rychlá implementace. Použití ve Vokabuláři webovém bude vypadat následovně - třída si vezme seznam repozitářů, ze kterých má být proveden import z hlavního importního manažera, a pro každý repozitář nechá skrze *ImportPipelineManager* vytvořit importní úlohy pro jednotlivé repozitáře.

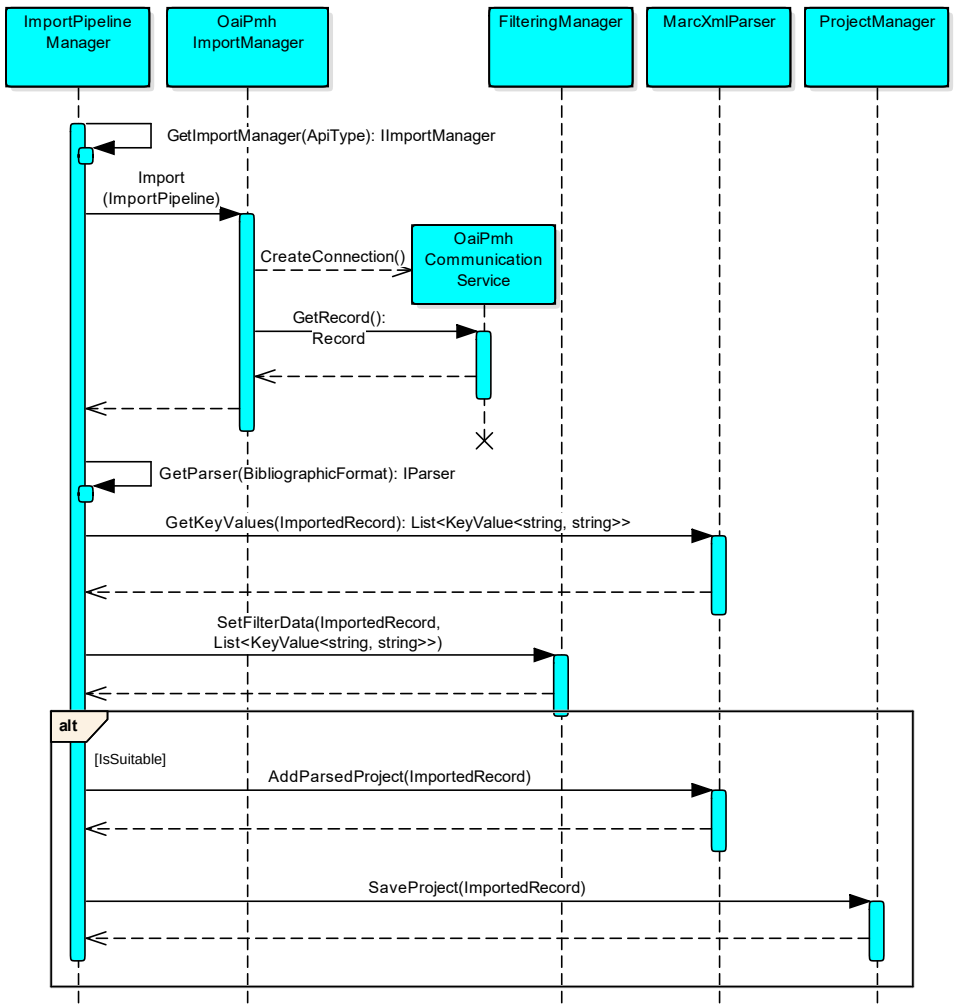


Obrázek 3.2: Diagram tříd - import ze zdroje

ImportPipelineManager nechá vytvořit objekt *ImportPipeline*, který se bude skládat z jednotlivých bloků činností. Objektu bude předána konfigurace externího repozitáře, která zajistí vybrání odpovídajícího konkrétního importního manažeru. Importní manažer se vybere podle typu rozhraní, které zdroj poskytuje. Následný průběh importu je zobrazen v diagramu 3.3.

Jednotlivé implementace importních manažerů budou odpovídat tomu, jaký typ rozhraní zdrojů dokážou zpracovat. Implementace budou používat již vlastní komunikační knihovny pro získání dat z daného zdroje. Dle získaných dat se vybere příslušný konvertor, který převede vstupní data na dvojice klíč - hodnota. Následně dojde k odeslaní dat do filtračního manažeru, kde bude rozhodnuto, jestli jsou vhodná pro uložení ve Vokabuláři webovém a mají se dále zpracovávat.

Data, která projdou filtračním manažerem se poté konvertují podle typu bibliografického formátu příslušnými konvertory do entit Vokabuláře webového. Entity budou následně uloženy do relační databáze.



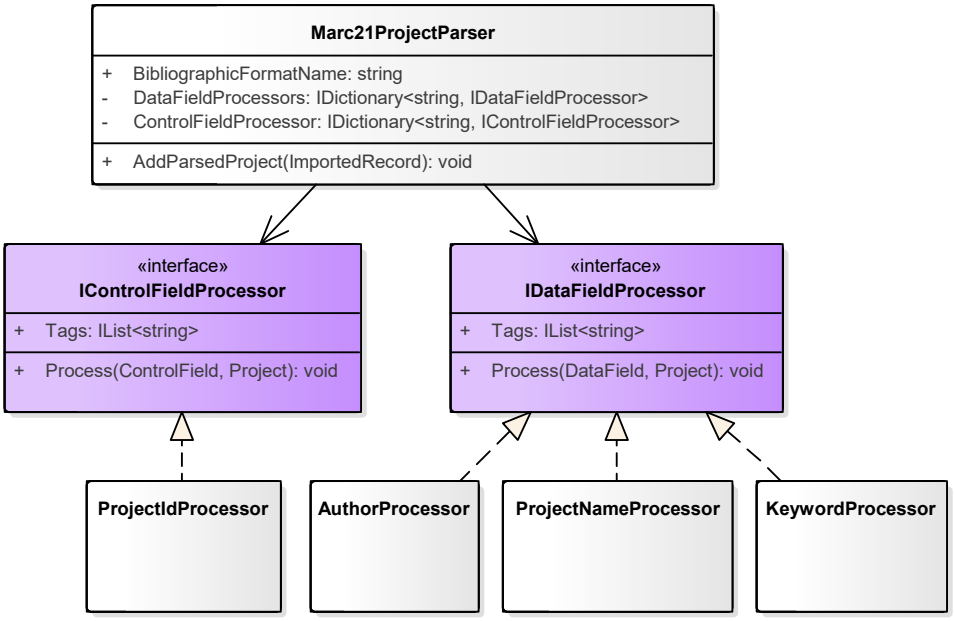
Obrázek 3.3: Sekvenční diagram - import ze zdroje

3.1.2 Komunikační klient OAI-PMH

Komunikační klient bude mít metody založené na klíčových slovech protokolu OAI-PMH. Klient poskládá URL dotaz, XML odpověď, kterou získá, převede na instanci třídy vygenerovanou pomocí nástrojů pro převod XSD souborů na třídy v jazyce C#. Vygenerována bude podle XSD souboru, který definuje XML odpověď protokolu OAI-PMH. Kromě této třídy budou vygenerovány i všechny ostatní třídy související s XML odpovědí protokolu.

3.1.3 MARC 21 konvertor

Architektura konvertoru MARC 21 je vyobrazena pomocí diagramu tříd na obrázku 3.4. Konvertor bude obsahovat procesory pro zpracování kontrolních a datových polí. Každý procesor bude zpracovávat pouze jednu logickou část metadat (např. autory, informace o vydání, název projektu). Z jakých polí tyto informace dokáže získat, určuje seznam tagů. Procesorům se předává importovaný záznam a také projekt, do kterého přidají jimi získaná metadata. Rozdělení do tříd je vhodné z důvodu přehlednosti a snadné rozšiřitelnosti, díky využití IoC.

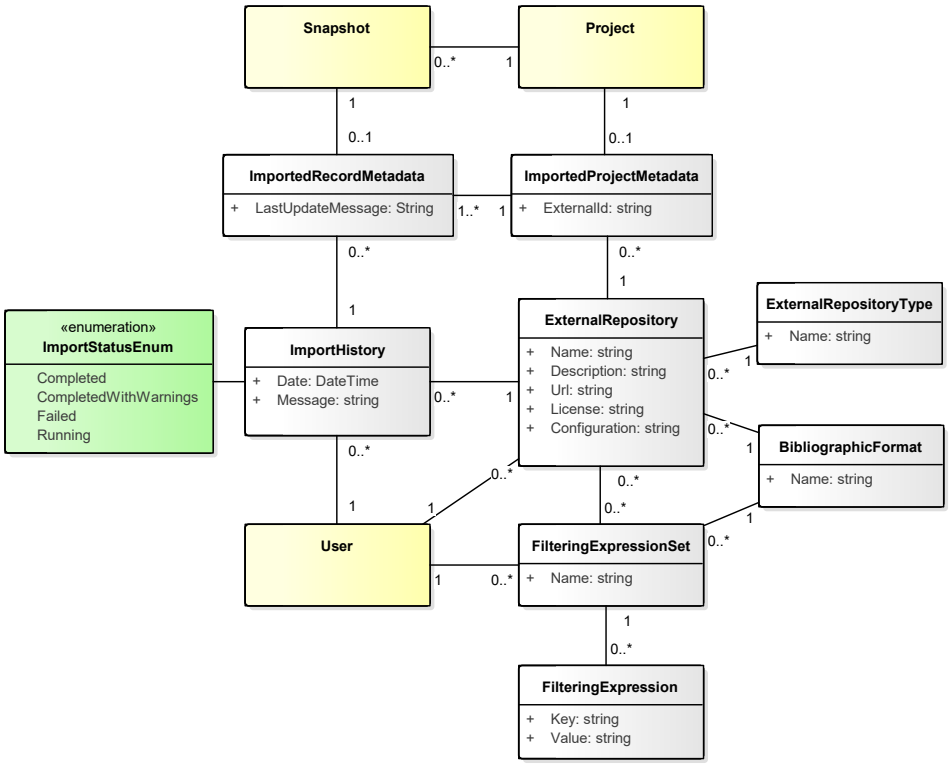


Obrázek 3.4: Diagram tříd - MARC 21 konvertor

3.2 Databázový model

Relační databázi Vokabuláře webového bude nutno rozšířit o tabulky vyobrazené na obrázku 3.5. Jedná se především o entity, které budou v sobě uchovávat konfiguraci zdrojů a historii importu. Tabulky *Project*, *Snapshot* uchovávají metadata o projektech, které se nachází ve Vokabuláři webovém, do těchto tabulek se budou ukládat data, které se zpracují pomocí příslušných konvertorů. V tabulce *Project* jsou uloženy základní informace o projektu – jeho název, kdy a kým byl vytvořen. Dále na sebe váže tabulky, které obsahují další informace o díle, např. klíčová slova, literární druh a žánr. Tabulka *Snapshot* uchovává verzovaná metadata o projektech. Tabulka *User* bude

v rozšíření především sloužit pro uchování, jaký uživatel dané entity založil. Tyto tři tabulky se již aktuálně nacházejí v relační databázi.



Obrázek 3.5: Databázový model

ExternalRepository

V této tabulce se uchovávají informace o zdroji, ze kterého lze provést import. Jaký typ rozhraní poskytuje daný zdroj, určuje tabulka *ExternalRepositoryType*, formát ve kterém jsou data poskytována se nachází v tabulce *BibliographicFormat*.

ImportHistory

Tabulka *ImportHistory* v sobě uchovává záznamy o proběhlých a běžících importech z daného externího repozitáře. Obsahuje informaci o uživateli, který import spustil, začátku importu, jeho stavu a případně chybovou hlášku, kvůli které byl ukončen. Vytváří se při zahájení importu z externího repozitáře. Váže na sebe tabulku *ImportedRecordMetadata*, která obsahuje verze záznamů, které byly v daném záznamu historie importovány.

ImportedRecordMetadata

Tato tabulka obsahuje sloupec, do kterého lze uložit chybovou hlášku, kvůli které nemohla být aktuálně zpracovávaná verze záznamu importována. V tabulce *Snapshot*, kterou na sebe váže, jsou uložena verzovaná konvertovaná metadata o jednotlivých projektech. O kterou verzi importovaného projektu se jedná, lze nalézt v tabulce *ImportedProjectMetadata*.

ImportedProjectMetadata

Uchovává dodatečné informace o importovaném záznamu, který se nachází v tabulce *Project*. Váže se na tabulku *ExternalRepository*, tudíž určuje, z jakého repozitáře byl záznam importován.

FilteringExpressionSet

Tato tabulka udává název skupiny filtračních výrazů (tabulka *FilteringExpression*), které na sebe váže. Jednotlivé filtrační výrazy se skládají z klíče a hodnoty. Možnosti filtrování budou následovné:

- Rovná se.
- Obsahuje.
- Začíná na.
- Končí na.

Dále se v tabulce *BibliographicFormat* se nachází informace, pro který bibliografický formát je určen tento set. Jeden set může být využíván více externími repozitáři a naopak – repozitář může mít více filtračních setů.

3.2.1 Stavy importu

Informace o tom, že započal import, se uloží při založení importní úlohy pro daný repozitář. V tu chvíli bude jeho status nastaven na probíhající. Pokud bude v průběhu importu na úrovni importní úlohy zachycena výjimka, import se ukončí a do databáze se uloží status neúspěšný import a chyba, kvůli které byl ukončen. Pokud bude chyba zachycena na úrovni jednotlivých bloků, uloží se informace do entity *ImportedRecordMetadata* a import bude pokračovat. Při dokončení importní úlohy se uloží stav importu jako úspěšný v případě, že žádná entita *ImportedRecordMetadata* neobsahuje chybu (v průběhu zpracování importní úlohy nebyla zachycena chyba na úrovni bloků). V opačném případě se nastaví stav importu jako úspěšný s varováním.

Při spuštění nového importu ze stejného repozitáře se získá poslední záznam importní historie, který skončil buď úspěšně, nebo úspěšně s varováním. Od toho data se začnou získávat modifikované záznamy skrze rozhraní.

3.3 Návrh GUI

Tato kapitola se zaměřuje na vizualizaci podoby integrace bibliografického modulu pro import z externí zdrojů do portálu Vokabuláře webového. Modul bude v portálu prezentován pomocí sekce *Import z externích repozitářů*, která bude dostupná pouze uživatelům s daným oprávněním. Dostat se do sekce bude možné skrze záložky administrace. Sekce bude rozčleněna do tří podsekcí s následujícími možnostmi:

- Externí repozitáře - seznam externích repozitářů, jejich tvorba, úpravy, mazání a zobrazování statistik importů z jednotlivých repozitářů.
- Import - spuštění importu ze seznamu externích repozitářů, sledování průběhu importu.
- Filtrační sety - seznam filtračních setů, jejich tvorba, úpravy, mazání.

Podsekce budou přístupné skrze postranní menu v levé části obrazovky.

Návrhy UI budou vytvořeny pomocí software Axure RP8 [37]. Aby byla zachována konzistence a jednotný vzhled portálu, vycházejí návrhy z aktuální podoby Vokabuláře webového. Horní menu je zcela převzato a formuláře jsou inspirovány již stávajícími formuláři, které se v portálu nacházejí. Seznamy mají stejný vzhled jako seznam v sekci Bibliografie.

3.3.1 Seznam externích repozitářů

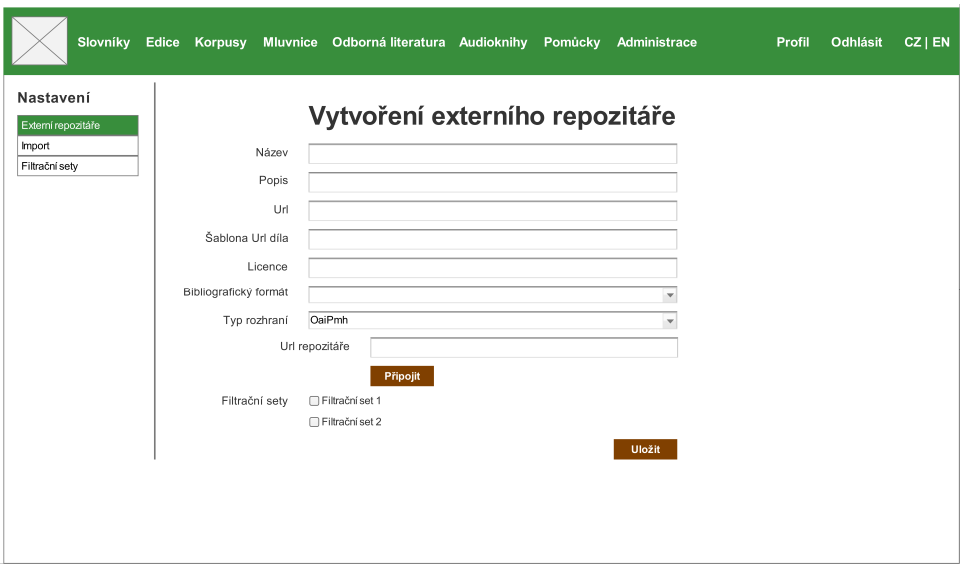
Obrazovka se seznamem externích repozitářů bude uživateli zobrazena jako první poté, co vstoupí do sekce pro správu importu. Její vizualizace se nachází na obrázku 3.6. U každého repozitáře budou kromě názvu a popisu i základní informace (např. typ rozhraní, bibliografický formát, licence). Dále zde bude panel, který bude umožňovat úpravu repozitáře, jeho smazání a také zobrazení detailu. V detailu se budou nacházet informace o posledním importu (např. kdy a kým byl proveden, kolik položek bylo importováno).

3.3.2 Vytvoření externího repozitáře

Formulář pro vytvoření externího repozitáře (obrázek 3.7) se skládá z části, která je stejná pro všechny repozitáře, a z části, která je dána typem rozhraní repozitáře. Jakmile bude vybrán *Typ rozhraní*, závislá část formuláře se získá pomocí technologie AJAX a vykreslí se. V případě vizualizace návrhu se jedná o pole *URL repozitáře* a tlačítko *Připojit*. Na konci formuláře se bude

nacházet seznam filtračních setů, kde bude možno vybrat ty sety, které se mají použít pro filtraci záznamů při importu z tohoto repozitáře.

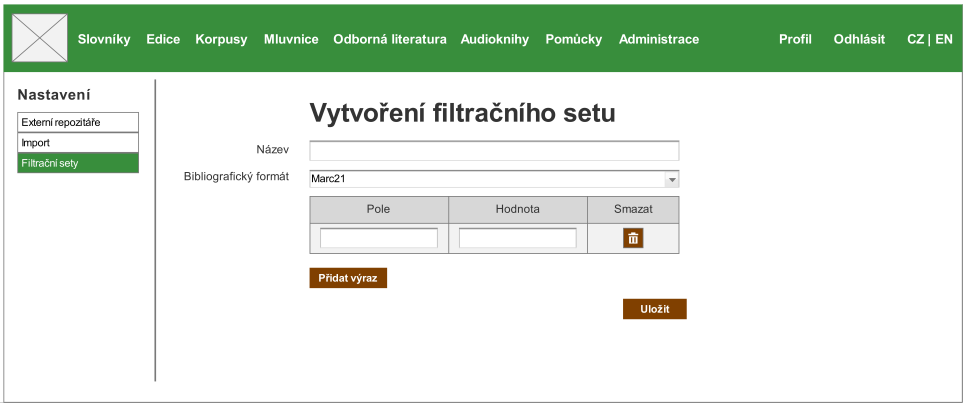
Obrázek 3.6: Seznam externích repozitářů



Obrázek 3.7: Vytvoření externího repozitáře

3.3.3 Vytvoření filtračního setu

Ve formuláři pro vytvoření filtračního setu (obrázek 3.8) se kromě názvu a typu formátu, pro který bude set určen, bude nacházet i dynamická tabulka, ve které lze přidávat jednotlivé filtrační výrazy. Přidání řádku pro další filtrační výraz, se bude provádět pomocí tlačítka *Přidat výraz*.



Obrázek 3.8: Vytvoření filtračního setu

Kapitola 4

Realizace

Tato kapitola se zabývá zejména technickými záležitostmi. Nachází se zde popis struktury projektů, implementace komunikační knihovny pro protokol OAI-PMH, knihovny zajišťující import a také problémy, které nastaly během implementace.

4.1 Struktura projektů

V rámci implementace došlo k vytvoření nových projektů, jimiž byla struktura Vokabuláře webového rozšířena. Názvy a funkce projektů jsou popsány v následujícím seznamu:

- *Vokabular.ProjectImport* – vstupní bod celého procesu importu, obsahuje logiku, jenž vybírá příslušné komunikační klienty a konvertory dat, filtruje importované záznamy a také se stará o uložení importovaných záznamů do relační databáze.
- *Vokabular.ProjectImport.Model* – definuje rozhraní a entity, jenž se využívají v rámci procesu importu.
- *Vokabular.ProjectImport.Shared* – obsahuje třídy, které jsou sdíleny napříč projekty (např. konstanty).
- *Vokabular.ProjectImport.Test* – slouží pro testování jednotlivých částí procesu importu pomocí jednotkových testů, ale i pro testování celého procesu pomocí integračních testů.
- *Vokabular.ProjectParsing.Model* – definuje rozhraní a entity, jenž se využívají v rámci konverze dat.
- *Vokabular.OaiPmhProjectImportManager* – knihovna pro komunikaci s rozhraním OAI-PMH.

- *Vokabular.Marc21ProjectParser* – obsahuje konverzi dat ze standardu MARC 21 ve formátu XML do entit Vokabuláře webového.
- *Vokabular.Marc21ProjectParser.Test* – slouží pro testování konverze dat z formátu MARC 21.

Kromě nově vytvořených projektů, bylo několik stávajících projektů Vokabuláře webového rozšířeno. Jednalo se zejména o integraci nových projektů do stávající struktury. Rozšířeny byly tyto projekty:

- *Vokabular.DataEntities* – zprostředkování přístupu k databázi, rozšířeno o nové entity.
- *Vokabular.MainService* – poskytuje REST API, rozšířeno o kontroly, které zpřístupňují manipulaci s entitami, spuštění importu a získávání informací o něm.
- *ITJakub.Web.Hub* – webový portál, rozšířeno a pohledy, jenž prezentují uživatelům rozhraní pro správu importu z externích repozitářů.

4.2 Implementace komunikačního klienta OAI-PMH

Komunikační klient OAI-PMH využívá třídu *HttpClient*, jenž se nachází ve frameworku ASP.NET Core k vytváření HTTP spojení s repozitářem, jenž vystavuje OAI-PMH rozhraní. Komunikační klient má metody založené na klíčových slovech protokolu OAI-PMH. Zejména to jsou následující metody:

- metoda pro získání informací o repozitáři (klíčové slovo *Identify*),
- metoda pro získání dostupných metadatových formátů (klíčové slovo *ListMetadataFormats*),
- metoda pro získání dostupných setů (klíčové slovo *ListSets*),
- metoda pro získání dostupných identifikátorů (klíčové slovo *ListIdentifiers*),
- metoda pro získání seznamu záznamů (klíčové slovo *ListRecords*),
- metoda pro získání záznamu (klíčové slovo *GetRecord*).

Pro metody s klíčovými slovy *ListSets*, *ListIdentifiers* a *ListRecords* jsou i dostupné metody, které budou přijímat jako argument *resumptionToken* a budou tím pádem vracet zbytek seznamu.

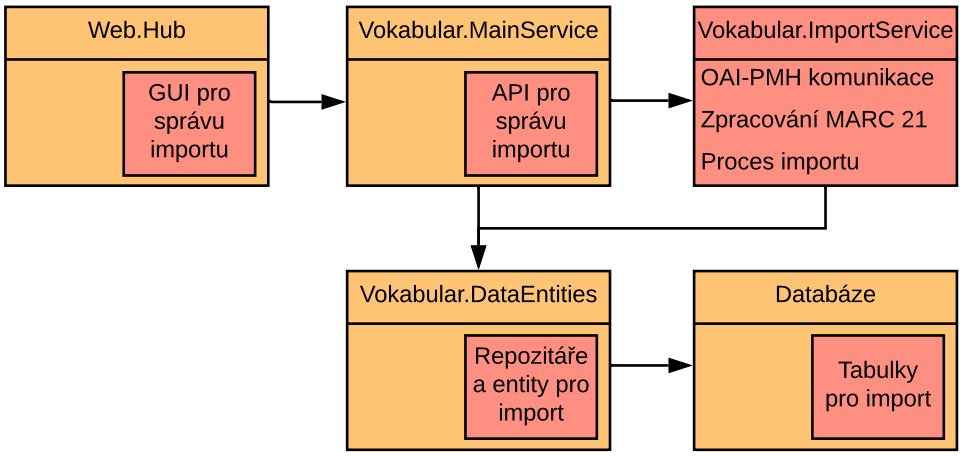
Tyto metody navážou pomocí třídy *HttpClient* spojení na URL adrese, kterou vytvoří dle zadaných parametrů. Například pro získání seznamu identifikátorů to bude příslušné klíčové slovo *ListIdentifiers* a typ požadovaného metadatového formátu (atribut *metadataPrefix*). Následně se vrátí odpověď ve formátu XML. Tato odpověď bude deserializována na instanci třídy *OaiPmhResponse*. Tato třída je vygenerována pomocí nástrojů pro převod XSD souborů na třídy v jazyce C#. Vygenerována je podle XSD souboru, který definuje XML odpověď protokolu OAI-PMH, kromě této třídy jsou vygenerovány i všechny ostatní třídy související s XML odpovědí protokolu.

Instance odpovědi *OaiPmhResponse* vrácené protokolem OAI-PMH se zkontroluje, zda neobsahuje chybu. V případě validní odpovědi se převede na požadovaný formát podle toho, o jakou metodu se jedná. V případě metody pro získání dostupných identifikátorů se odpověď převede na seznam identifikátorů.

V průběhu implementace bylo zjištěno, že server Historického ústavu AV ČR, který vystavuje rozhraní OAI-PMH, není úplně stabilní, a proto musely být přidány mechanismy, jenž řeší výpadky tohoto serveru. Při získávání seznamu záznamů pomocí tokenu se v případě obdržení HTTP chybového kódu dotaz na server opakuje se zpožděním, které je nastaveno v konfiguračním souboru a exponenciálně se zvyšuje při dalším nepovedeném dotazu. Dále je v tomto souboru nastaveno, kolikrát se má dotaz maximálně opakovat. Pokud je překročen maximální počet opakování, je import přerušen.

4.3 Integrace do Vokabuláře webového

Integrace do Vokabuláře webového byla provedena na několika úrovní. Její vizuální zobrazení se nachází na obrázku 4.1.



Obrázek 4.1: Model integrace komponent

Jak již bylo zmíněno v kapitole 4.1 Struktura projektů, projekt *Vokabular.ProjectImport* je vstupním bodem pro proces importu. Tento projekt je napojen do projektu *Vokabular.MainService*, který zpřístupňuje import pomocí REST API jiným aplikacím. Jednou z těchto aplikací je projekt ITJakub.Web.Hub, který běží na webovém serveru. Ten byl rozšířen o následující obrazovky (náhledy obrazovek se nachází v příloze B):

- Seznam externích repozitářů.
- Formulář pro přidání nebo úpravu externího repozitáře.
- Seznam externích repozitářů, u kterých lze provést import.
- Průběh importu z daných repozitářů.
- Seznam filtračních setů.
- Formulář pro přidání nebo úpravu filtračního setu.

Další částí, kde došlo k rozšíření, byl datový model relační databáze Vokabuláře webového. Úpravy struktury byly provedeny pomocí vytvoření SQL skriptů, jenž přidají nové tabulky a sloupce do aktuální struktury databáze. V projektu *Vokabular.DataEntities* jsou doplněny a upraveny mapovací soubory pro ORM framework NHibernate.

4.4 Přístup do relační databáze

Jak již bylo zmíněno v návrhu, proces importu se skládá z jednotlivých bloků. V implementaci byla využita knihovna Task Parallel Library [38]. Ta obsahuje komponenty pro řízení toku dat, jenž umožňují i snadnou paralelizaci zpracování dat v toku. To vedlo k problému s přístupem do databáze. Ve Vokabuláři webovém je přístup do databáze obstaráván pomocí návrhového vzoru Unit Of Work, jehož implementace je v IoC kontejneru zaregistrována jako *scoped*. Tudíž jedna a ta samá instance této třídy je dostupná pro všechny bloky importu. Těmto blokům umožňovala vytvářet a uzavírat databázové transakce, v čemž nastal problém. Pokud jeden blok chtěl ukládat data, vytvořil transakci a začal s přípravou dat pro uložení. Mezitím požádal další blok o transakci do databáze, tudíž mu byla předána již vytvořená transakce. Následně první blok uložil data a uzavřel transakci, což mělo za následek, že druhý blok při ukládání již neměl platnou transakci.

Řešením bylo pro každý blok vždy vytvořit tzv. *IServiceScope* skrze *IServiceProvider*. Každý *IServiceScope* poté poskytuje své instance tříd registrovaných jako *scoped*, tudíž každý blok má svůj vlastní Unit Of Work, který mu bude poskytovat transakce, které již nebudou sdílené s ostatními bloky [39].

Kapitola 5

Testování

Nedílnou součástí práce je testování. Vzhledem k tomu, že se jedná zejména o knihovnu pro import a konverzi dat, byly nejprve provedeny unit testy, které cílily na otestování jednotlivých menších částí implementace. Následovaly integrační testy, které se zaměřily na již větší celky. Na závěr byly provedeny testy nad reálnými daty.

5.1 Unit testy

Pomocí Unit testů byly otestovány části proces importu a konverzní knihovny pro formát MARC 21. V rámci toho byly vytvořeny dva testovací projekty *Vokabular.ProjectImport.Test* a *Vokabular.Marc21ProjectParser.Test*.

5.1.1 Nastavení prostředí

Jako testovací framework byl zvolen MSTest od společnosti Microsoft, který je použitý i ve zbytku Vokabuláře webového. Zvolen byl z důvodu jednotnosti použití testovacího frameworku v celém Vokabuláři webovém.

Pro testování samostatných jednotek bez vlivu implementace ostatních tříd, na kterých je testovaná třída závislá, bylo nutné zajistit mockování. Pro pohodlné mockování byla zvolena knihovna Moq ve verzi 4, která umožňuje realizovat rozhraní či nahradit funkcionalitu třídy jinou implementací. V případě, že testovaná třída přijímá v konstruktoru instance tříd, které potřebuje ke svému běhu, nevkládají se instance z IoC kontejneru, ale vkládají se namockované objekty, které zajistí správné testovací prostředí.

Jelikož se testuje i uložení dat do databáze, je zde využitu SQLite databáze. Databáze je v zde využívána v tzv. in-memory database režimu, který uchovává

databázi v operační paměti počítače. Schéma SQLite databáze se vytváří podle mapovacích souborů, jenž se nachází v projektu *Vokabular.DataEntites*. Schéma se vytváří pro každý test znovu, tím je zajištěno, že se v databázi nenachází nechtěná data, která by mohla ovlivnit výsledek probíhajícího testu.

5.1.2 Průběh testování

V projektu *Vokabular.ProjectImport.Test* se nachází několik testovacích tříd, jenž testují různé části procesu importu. První jsou třídy, jenž testují filtrační manažer. Filtrační manažer je nejprve testován z pohledu filtrovaní dle zadaných filtrů (rovná se, začíná na, atd.). Další testy se zaměřují na filtrování podle získaných metadat, např. dle toho, jestli záznam smazaný, případně zda časová známka záznamu je novější, než časová známka záznamu uloženého v databázi (kontroluje se pouze v případě, že záznam byl úspěšně importován v předešlých importech).

Dále se testuje stavba objektu *ImportPipeline*. Testováno je, jestli se staví ve správném pořadí. Následně se testuje, jestli se bloky provolají ve správném pořadí a zda je i následné provolání jednotlivých bloků ve správném pořadí.

Poslední testovanou třídou v projektu *Vokabular.ProjectImport.Test* je třída, která zabývám ukládáním a aktualizováním importovaných projektů v relační databázi. Test ukládání se provádí z důvodu jeho komplexnosti – nachází se zde mnoho entit, které musí být uloženy korektně. První je testováno samotné vytvoření záznamu, další testy jsou zaměřeny na aktualizaci záznamu a to buď na úpravu stávajících metadat, nebo na přidání nových metadat k záznamu. Oba případy vedou na vytvoření nové verze záznamu.

Projekt *Vokabular.Marc21ProjectParser.Test* se zabývá testováním knihovny pro konverzi dat z formátu MARC 21. Testuje, zda procesory dokáží vstupní XML správně konvertovat na entity Vokabuláře webového a zda se dokáží procesory vypořádat s chybou.

5.2 Integrační testy

Kromě unit testů se v projektu *Vokabular.ProjectImport.Test* se nachází i integrační testy, které se zabývají testováním větších celků. Zejména se jedná o testování celého průběhu importu a paralelního průběhu importů z více repozitářů.

5.2.1 Nastavení prostředí

Stejně jako u unit testů je i zde využito testovacího frameworku MSTest, mockovací knihovny Moq a databáze SQLite.

V těchto testech se využívá i IoC, kde mockované objekty nahrazují již registrované třídy v IoC kontejneru. Opět se využívá SQLite databáze, ale zde již dochází i k vytváření mockovaných dat (externí repozitář, bibliografické formáty, atd.), které se před každým testem ukládají do databáze. Při každém testu se databáze vytváří znovu, což vede k bezproblémovému nastavení mockovaných dat, bez ohledu na výsledek předchozích testů.

5.2.2 Průběh testování

První testovací třída se zabývá uložením importovaného projektu a metadat o něm. Kromě testů, kde se sleduje úspěšné uložení dat, se zde nachází i testy, které testují, zda když při uložení importovaného projektu nastane chyba a jestli je tato informace zaznamenána v metadatech o importovaném projektu.

Další testovací třída již testuje celý proces importu od vložení repozitáře do hlavního importního manažeru až po vytvoření projektu. První testy ověřují vyvolání výjimky při pokusu zapnutí importu, bez dodání repozitářů, ze kterých se má provést import. Dále se testuje import z jednoho repozitáře. Nejprve je import otestován na jednom záznamu, poté je testováno, že při dodání dvou záznamů se navzájem neovlivňují. Na závěr je provedeno testování importu ze dvou repozitářů najednou. Zde se kontroluje, že nedochází k ovlivňování jednotlivých importů, a to ani v případě, že při jednom z importů nastane výjimka.

5.3 Testy s reálnými daty

Testování s reálnými daty bylo plánováno provést se všemi třemi externími bibliografickými databázemi, které byly zmíněny v kapitole 2.4. Nicméně kolegům z ÚJČ, kteří zajišťují provoz Vokabuláře webového, se zatím nepovedlo získat přístup ke všem vyjmenovaným, proto bylo testování provedeno pouze na bibliografické databázi Historického ústavu AV ČR.

5.3.1 Nastavení prostředí

Tyto testy byly prováděny importem dat z externí bibliografické databáze Bibliografie dějin Českých zemí[1], kterou spravuje Historický ústav AV ČR.

Data z bibliografické databáze byla získávána ve formátu MARCXML pomocí OAI-PMH rozhraní. Import byl spouštěn skrze nově vytvořené rozhraní pro správu importu, které se nachází na portálu Vokabuláře webového. Kromě webového portálu musela být spuštěna i hlavní služba Vokabuláře webového.

Test probíhal na stolním počítači s operačním systémem Windows 10 Pro x64, operační pamětí 16 GB a procesorem Intel Core i5-2400 CPU 3,10 GHz. K měření času byla použita knihovna System.Diagnostics.

5.3.2 Průběh testování

Testování bylo rozděleno na dvě části. Nejprve byla otestována rychlost, kterou externí bibliografická databáze vrací data ke zpracování. Následně byla otestována rychlost zpracování záznamů ve Vokabuláři webovém.

Testování externí bibliografické databáze

Otestováno bylo šest sad dat, kde pro každou sadu byly provedeny tři měření. Tento počet měření byl zvolen z důvodu eliminace chyby měření, která mohla nastat vnějšími vlivy. Různých velikostí sad bylo docíleno změnou parametrů dotazu, a to konkrétně doby, z jaké se mají záznamy importovat.

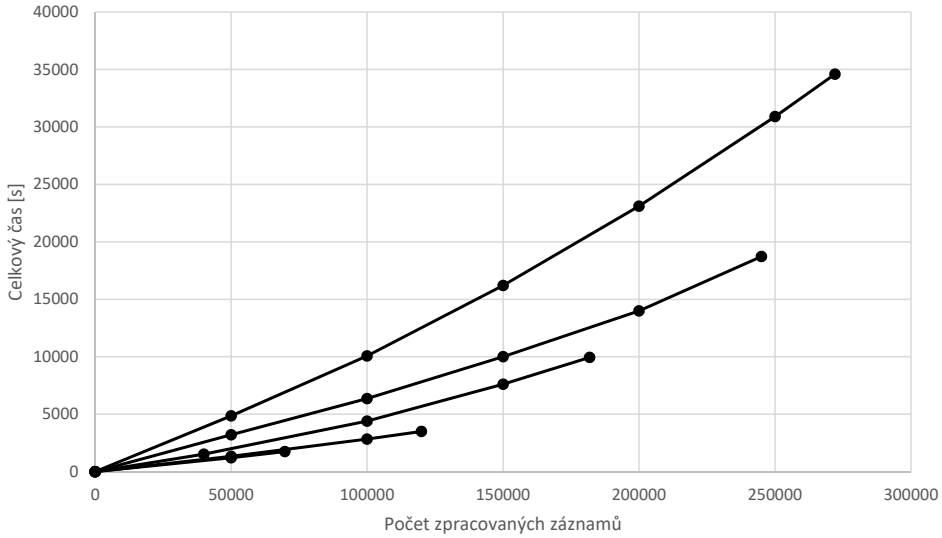
V následující tabulce 5.1, jsou vyobrazena naměřená data. Každý řádek odpovídá jedné sadě dat s již průměrnými hodnotami získaných zmíněnými třemi měřeními.

Z naměřených hodnot lze vyvodit, že s lineárně rostoucím počtem dat exponenciálně stoupá doba za kterou externí bibliografická databáze vrátí veškerá data, tudíž se rychlost snižuje. Na obrázku 5.1 je vyobrazen průběh importu pro jednotlivé sady dat a je zde vidět postupné zvyšování času na vrácení části záznamů v průběhu testování jednotlivých sad dat.

[1] Databáze je dostupná na adrese: `https://biblio.hiu.cas.cz/`.

Počet záznamů	Celkový průměrný čas [s]	Průměrný čas na 50 záznamů [s]
33700	564	0,84
69700	1746	1,25
120000	3612	1,49
181800	9942	2,73
245200	18720	3,82
272000	34584	6,35

Tabulka 5.1: Výsledky testování rychlosti odpovědí serveru



Obrázek 5.1: Vizualizace rychlosti odpovědí serveru

Testování rychlosti zpracování záznamů

Rychlost zpracování záznamů ve Vokabuláři webovém byla testována na importu 272000 záznamů, získaných z externí bibliografické databáze. Zpracováním záznamů se v tomto případě myslí konvertování záznamu z přijatého formátu MARC 21 do entit Vokabuláře webové a jejich následné uložení do relační databáze. Měření probíhalo postupně na jednotlivých částech seznamu záznamů, které vrací externí bibliografická databáze, tj. 50 záznamů. Testováním bylo zjištěno, že se záznamy zpracovávají průměrnou rychlostí 50 záznamů za 2,4 sekundy.

5.3.3 Výsledek testování

Při testování bylo zjištěno, že rychlost importu při menším počtu závisí na rychlosti zpracování dat ve Vokabuláři webovém. S rostoucím počtem dat pak přechází závislost na rychlost externí bibliografické databáze. Tento testovaný případ je nejhorší možný. Při reálném importu bude rychlost ovlivněna i filtrováním záznamů a jejich rozložení v rámci externí bibliografické databáze.

Kapitola 6

Závěr

Cílem této práce byla integrace externích bibliografických zdrojů do Vokabuláře webového. Nejprve byla provedena analýza protokolů OAI-PMH, Z39.50 a metadatových formátů MARCXML a Dublin Core, jenž poskytují externí bibliografické databáze, ze kterých se budou získávat metadata o dílech. Z analýzy vyplynulo, že pro komunikace je nejvhodnější použít protokol OAI-PMH, jelikož podporuje periodické sklízení dat a zároveň je dostupný u všech analyzovaných externích zdrojů. Jelikož nebyla nalezena žádná knihovna kompatibilní s Vokabulářem webovým, bylo nutné knihovnu implementovat. Z dostupných metadatových formátů byl zvolen MARCXML, jelikož má oproti formátu Dublin Core podrobnější a přesně definovanou strukturu dat podle bibliografického standardu MARC 21.

Následně byl proveden návrh knihovny pro import s architekturou umožňující přidávání dalších komunikačních protokolů a metadatových formátů. Navrhnuty byly i knihovny pro komunikaci pomocí protokolu OAI-PMH, konverzi dat z formátu MARCXML a import dat do Vokabuláře webového. Navržena byla i část GUI pro portál Vokabuláře webového, který umožňuje správu importu.

Dle těchto návrhů byly knihovny implementovány a následně integrovány do Vokabuláře webového, čímž byly splněny všechny definované požadavky. Integrace proběhla na hlavní službě Vokabuláře webového, která nyní využívá knihovnu pro import dat z externích bibliografických zdrojů. Také byla rozšířena databáze o entity, které využívá importní knihovna. Další rozšíření proběhlo v aplikaci *ITJakub.Web.Hub*, kde bylo rozšířeno uživatelské rozhraní o možnost správy importu.

Zároveň byla implementace testována. Testování proběhlo zprvu pomocí unit testů, které testovaly jednotlivé části kódu. Následovaly je integrační testy, které se zaměřily na větší celky. Na závěr proběhly testy s reálnými daty, kde se měřila rychlost importu.

V budoucnu je možné rozšířit Vokabulář webový o možnost přidávání referencí k projektům na díla, která jsou podobná nebo se zaobírají stejným tématem. Reference na díla by bylo možné přidávat při tvorbě projektu ve Vokabuláři webovém. Reference by mohly být uváděny jak na díla, která byla vytvořená ve Vokabuláři webovém, tak i na díla importovaná skrze modul vytvořený v této práci.

Příloha A

Literatura

[1] ČERNÁ, Alena a Boris LEHEČKA. *Vokabulář webový* [online]. oddělení vývoje jazyka, Ústav pro jazyk český AV ČR, v. v. i. [cit. 2018-11-06]. Dostupné z: `http://vokabular.ujc.cas.cz/informace.aspx?t=ovokabulari&o=ovokabulari`

[2] *Informační technologie ve službách jazykového kulturního bohatství (IT JAKUB)* [online]. Ústav pro jazyk český AV ČR, v. v. i., 2015 [cit. 2018-11-06]. Dostupné z: `http://www.ujc.cas.cz/veda-vyzkum/vyzkum/grantove-projekty-ukoncene/informacni-technologie-it-jakub.html`

[3] *Výzkumná infrastruktura pro diachronní bohemistiku* [online]. Ústav pro jazyk český AV ČR, v. v. i., 2016 [cit. 2018-11-06]. Dostupné z: `http://vokabular.ujc.cas.cz/informace.aspx?t=ridics&o=ovokabulari`

[4] TROELSEN, Andrew a Philip JAPIKSE. *Pro C# 7: With .NET and .NET Core*. Apress, 2017. ISBN-13: 978-1484230176.

[5] FOWLER, Martin. Model View Controller. In: *martinfowler.com* [online]. Martin Fowler, 2005 [cit. 2019-02-25]. Dostupné z: `https://martinfowler.com/eaaDev/uiArchs.html`

[6] *Razor syntax reference for ASP.NET Core* [online]. Microsoft, 2018 [cit. 2019-02-25]. Dostupné z: `https://docs.microsoft.com/cs-cz/aspnet/core/mvc/views/razor?view=aspnetcore-2.2`

[7] FOWLER, Martin. Inversion of Control. In: *martinfowler.com* [online]. Martin Fowler, 2005 [cit. 2019-01-06]. Dostupné z: `https://martinfowler.com/bliki/InversionOfControl.html`

[8] *SQL Server Documentation* [online]. Microsoft, 2018 [cit. 2019-04-16]. Dostupné z: `https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017`

[9] FENTON, Steve. *Pro TypeScript: Application-Scale JavaScript Development. The expert's voice in TypeScript.* Apress, 2014. ISBN-13: 978-1-4302-6790-4.

[10] The Core Less Team. *Less.js* [online]. Less [cit. 2019-03-05]. Dostupné z: `http://lesscss.org/`

[11] *Open Archives Initiative - Protocol for Metadata Harvesting - v.2.0* [online]. Open Archives Initiative [cit. 2018-11-09]. Dostupné z: `https://www.openarchives.org/OAI/openarchivesprotocol.html`

[12] VAN DE SOMPEL, Herbert a Carl LAGOZE. The Santa Fe Convention of the Open Archives Initiative. In: *D-Lib Magazine* [online]. 2000, 6(2) [cit. 2018-11-08]. ISSN 1082-9873. Dostupné z: `http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html`

[13] THANOS, Constatino. *Research and Advanced Technology for Digital Libraries: 6th European Conference, ECDL 2002, Rome, Italy, September 16-18, 2002, Proceedings* Springer Berlin Heidelberg, 2003. 151-157. ISBN 978-3-540-45747-3.

[14] *EThOS as a data provider* [online]. British Library [cit. 2018-11-09]. Dostupné z: `https://www.bl.uk/ethos-and-theses/ethos-as-a-data-provider`

[15] DEVARAKONDA, Ranjeet, Giri PALANISAMY, James GREEN a Bruce E. WILSON, *Mercury: An Example of Effective Software Reuse for Metadata Management, Data Discovery and Access* [online]. ResearchGate, 2008 [cit. 2018-11-09]. Dostupné z: `https://www.researchgate.net/publication/234354172`

[16] *Využití protokolu OAI-PMH — Souborný katalog ČR - Portál CASLIN* [online]. Národní knihovna ČR [cit. 2018-11-09]. Dostupné z: `https://www.caslin.cz/caslin/spoluprace/jak-prispivat-do-sk-cr/dodavani-dat/vyuziti-protokolu-oai-pmh`

[17] *DCMI: Dublin Core Metadata Element Set, Version 1.1: Reference Description* [online]. The Dublin Core Metadata Initiative [cit. 2018-11-09]. Dostupné z: `http://dublincore.org/documents/dces/`

[18] *Connecting digital architectural archives with MACE - Metadata for Architectural Contents in Europe* [online]. ResearchGate GmbH [cit. 2018-11-09]. Dostupné z: `https://www.researchgate.net/publication/258241692`

[19] BARTOŠ, Ivan a Bohdan ŠMILAUER. Úvod do protokolu Z39.50 [online]. Státní technická knihovna [cit. 2018-12-22]. Dostupné z: `https://old.stk.cz/ZIG/UvodDoZ3950.doc`

[20] RUBRINGER, Tomáš. Z39.50. Ikaros [online]. 1999, ročník 3, číslo 8 [cit. 2018-12-23].ISSN 1212-5075. Dostupné z: http://ikaros.cz/node/10989

[21] MICHAEL, J. James a Mark HINNEBUSCH. *From A to Z39.50: A Networking Primer (Supplement to computers in libraries)*. Mecklermedia Corporation, c1995. ISBN-13: 978-0887367663.

[22] *Dublin Core* [online]. Dublin Core Metadata Initiative, 2019 [cit. 2018-12-10]. Dostupné z: http://dublincore.org/specifications/dublin-core/

[23] *DCMI: DCMI Qualifiers* [online]. Dublin Core Metadata Initiative, 2000 [cit. 2018-12-12]. Dostupné z: http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/

[24] *DCMI: Dublin Core Metadata Terms* [online]. Dublin Core Metadata Initiative, 2005 [cit. 2018-12-10]. Dostupné z: http://www.dublincore.org/documents/2005/06/13/dcmi-terms/

[25] *Dublin Core Czech Homepage* [online]. Ústav výpočetní techniky Masarykovy univerzity, 2006 [cit. 2018-12-10]. Dostupné z: http://webserver.ics.muni.cz/dublin_core/elems.html

[26] *The MARC 21 Formats: Background and Principles* [online]. Library of Congress, 1996 [cit. 2019-01-01]. Dostupné z: https://www.loc.gov/marc/96principl.html.

[27] *MARC 21 Format for Bibliographic Data : Introduction* [online]. Network Development and MARC Standards Office, Library of Congress, 2006 [cit. 2018-12-21]. Dostupné z: https://www.loc.gov/marc/bibliographic/bdintro.html

[28] OPPELT, Šimon. *Výměnné formáty v České republice se zaměřením na důvody přechodu z výměnného formátu UNIMARC na výměnný formát MARC 21* [online]. Praha, 2013 [cit. 2018-12-21]. Dostupné z: https://is.cuni.cz/webapps/zzp/detail/106965. Bakalářská práce. Karlova univerzita, Filozofická fakulta. Vedoucí práce Klára Rösslerová

[29] *Formát MARC21: čím se liší od formátu UNIMARC* [online] Národní knihovna České republiky, 2014 [cit. 2018-12-21]. Dostupné z https://nkp.cz/o-knihovne/odborne-cinnosti/zpracovani-fondu/informativni-materialy/marc21

[30] STODOLA, Jiří. Pravidla a formáty. In: *Sémantické aspekty katalogizace* [online]. Masarykova univerzita, 2014 [cit. 2018-12-30]. Dostupné z: https://is.muni.cz/do/rect/el/estud/ff/js14/katalogizace/web/pages/09-pravidla-a-formaty.html

[31] *Přehled polí a podpolí MARC21* [online] Národní knihovna České republiky, 2012 [cit. 2018-12-31]. Dostupné z: http://text.nkp.cz/o-knihovne/odborne-cinnosti/zpracovani-fondu/roztridit/stt-prehled

[32] Obrázek 9.1. In: *Sémantické aspekty katalogizace* [online]. Masarykova univerzita, 2014 [cit. 2018-12-30]. Dostupné z: https://is.muni.cz/do/rect/el/estud/ff/js14/katalogizace/web/pages/09-pravidla-a-formaty.html

[33] *MARC XML* [online]. Library of Congress, 2004 [cit. 2019-01-01]. Dostupné z: http://www.loc.gov/standards/marcxml/marcxml-overview.html.

[34] KADRNOŽKOVÁ, Eva. *Využití jazyka XML v knihovnické praxi* [online]. Brno, 2006 [cit. 2019-01-01]. Dostupné z: https://is.muni.cz/th/xuqhi/. Diplomová práce. Masarykova univerzita, Filozofická fakulta. Vedoucí práce Miroslav Bartošek.

[35] HTTP request methods. In *MDN web docs* [online]. Mozilla and individual contributors, 2019 [cit. 2019-04-21]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[36] *Background tasks with hosted services in ASP.NET Core* [online]. Microsoft, 2019 [cit. 2019-04-21]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-2.2

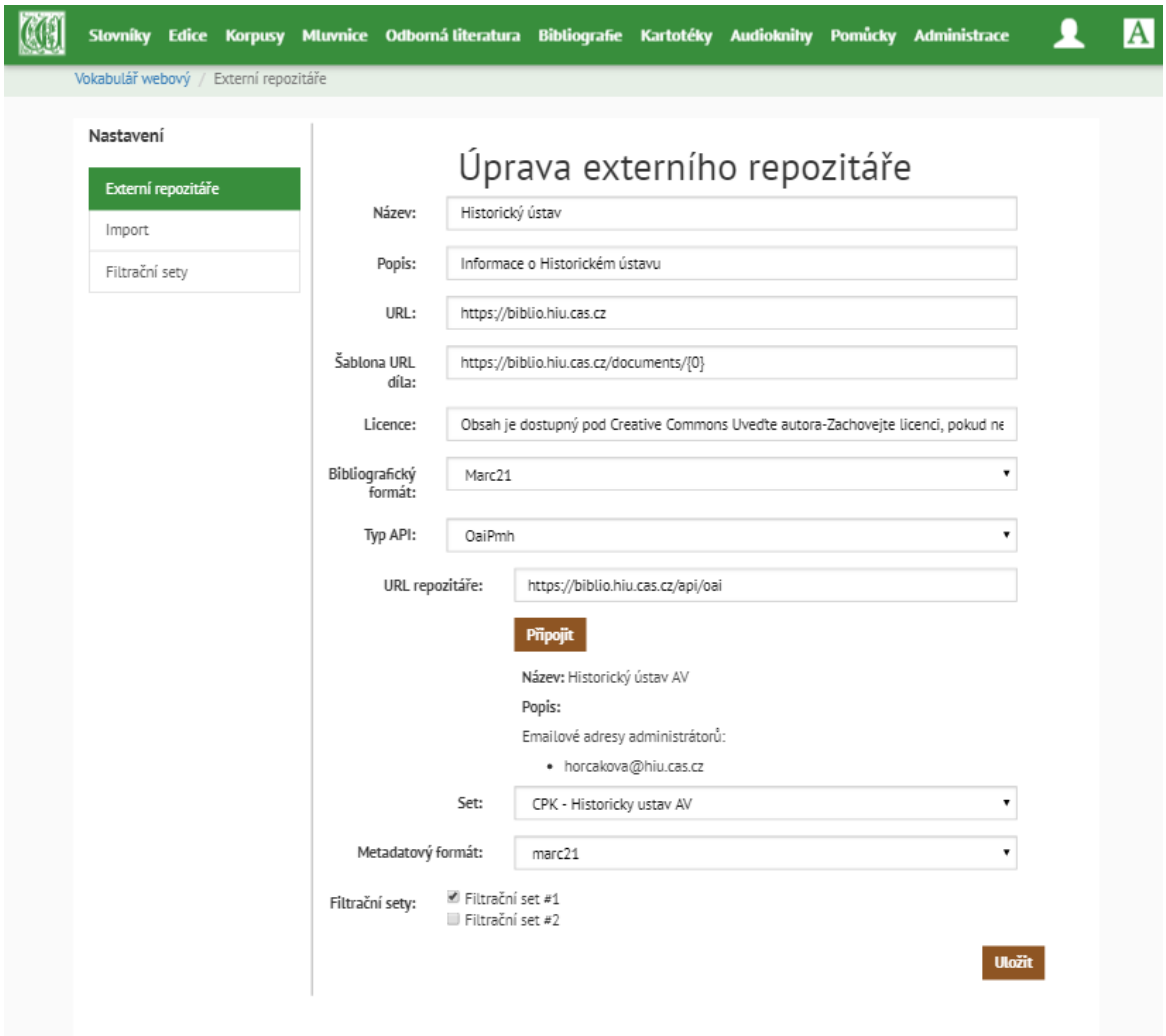
[37] *Prototypes, Specifications, and Diagrams in One Tool - Axure* [online]. Axure Software Solutions, 2019 [cit. 2019-04-21]. Dostupné z: https://www.axure.com

[38] *Dataflow (Task Parallel Library)* [online]. Microsoft, 2019 [cit. 2019-05-06]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/standard/parallel-programming/dataflow-task-parallel-library

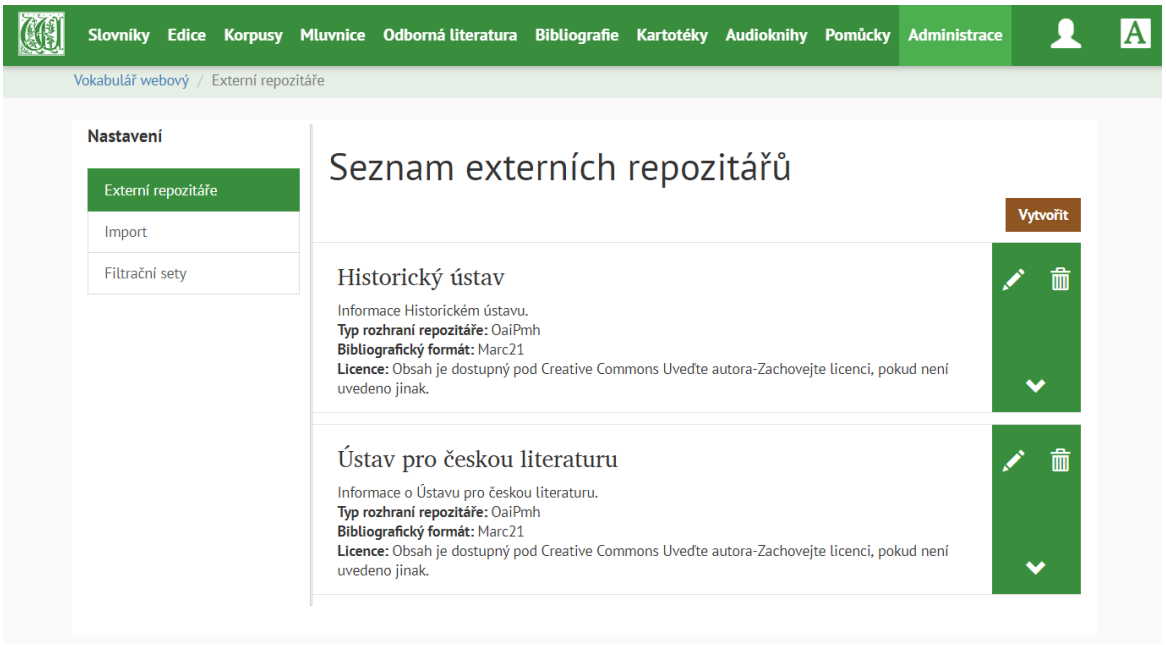
[39] *Dependency injection in ASP.NET Core* [online]. Microsoft, 2019 [cit. 2019-05-13]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.2

Příloha B

Náhledy implementovaného GUI

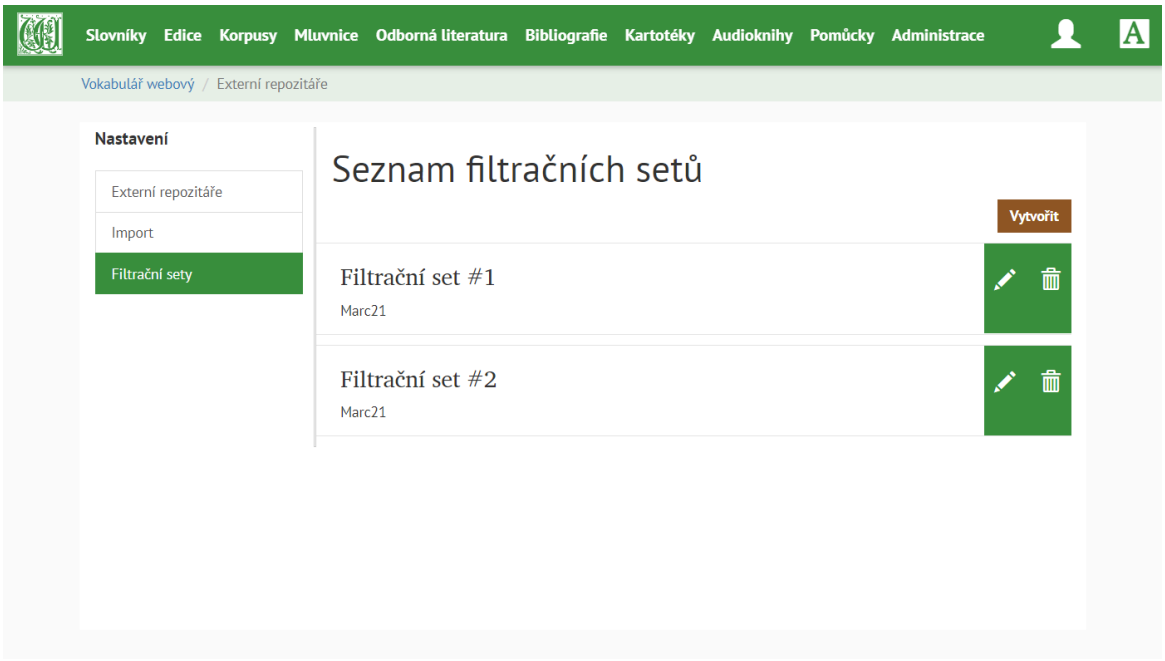


Obrázek B.1: Obrazovka – úprava repozitáře

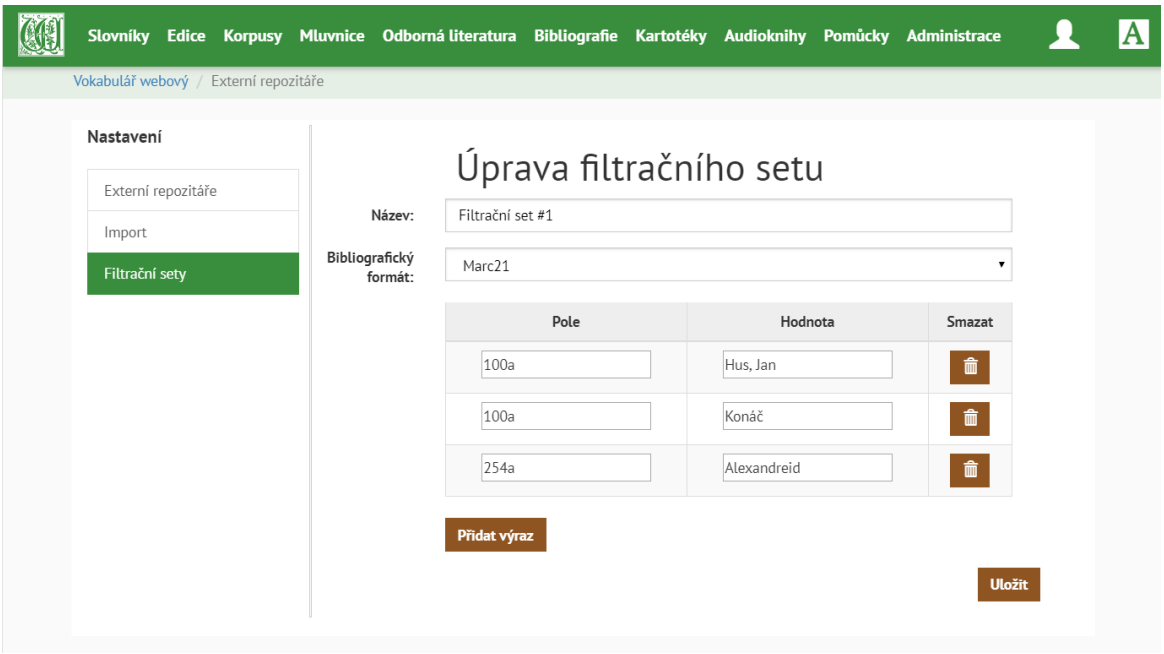


Obrázek B.2: Obrazovka – seznam repozitářů

Obrázek B.3: Obrazovka – spuštění importu



Obrázek B.4: Obrazovka – list filtračních setů



Obrázek B.5: Obrazovka – úprava filtračního setu

Příloha C

Seznam použitých zkratek

API	Application Programming Interface
AV ČR	Akademie věd České republiky.
DC	Dublin Core
DCMI	Dublin Core Metadata Initiative
HTTP	Hypertext Transfer Protocol
IoC	Inversion of Control
MARC	MAchine-Readable Cataloging
OAI	Open Archives Initiative
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
ORM	Object-relational mapping
SK ČR	Souborný katalog České republiky
GUI	Graphical User Interface
ÚJČ	Ústav pro jazyk český AV ČR, v. v. i.
UNIMARC	Universal MARC
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition