Mendelova univerzita v Brně
Provozně ekonomická fakulta

Studie katalogizačního systému pro malé a střední knihovny

Diplomová práce

Vedoucí práce:
Ing. Michael Štencl, Ph.D.

Bc. Radek Majer

Brno 2012

ZADANÍ PRÁCE

Rád bych touto cestou poděkoval Ing. Michaelovi Štenclovi, Ph.D. za veškerou pomoc, cenné rady a připomínky, které mi poskytl při zpracování diplomové práce. Dále bych rád poděkoval firmám LANius, s.r.o. a Effectiva Solutions, s.r.o., bez kterých by tato práce nevznikla. Na závěr patří velké poděkování celé mojí rodině za trvalou podporu během mého studia.

Prohlašuji, že tuto diplomovou práci vytvořil sám s použitím uvedených zdrojů.

V Brně, dne 10. května 2012 .....

Abstract

Majer, R. *The study of cataloging system for small and medium libraries*. Diploma thesis. Brno, 2012.

This diploma thesis deals mainly with pre-implementation stages of development of a new cataloging system. These are market research, selecting the target market segment and competitive analysis. In the target segment will be examined the requirements and then will be performed the proposal of the basis system and selected sub-system parts. The resulting work will serve as the basis for emerging cataloging system.

Keywords

library systems, multi-library systems, web appliactions, FRBR, OAI-PMH, MARC21, Z39.50, GWT, Ujorm

Abstrakt

Majer, R. *Studie katalogizačního systému pro malé a střední knihovny. Diplomová práce. Brno, 2012*. Diplomová práce. Brno, 2012.

Diplomová práce se zabývá zejména předimplementační fází vývoje nového katalogizačního systému. Jedná se o průzkum trhu, volbu cílového tržního segmentu a analýzu konkurence. V cílovém segmentu budou prozkoumány požadavky a na jejich základě bude proveden návrh celého systému a vybrané dílčí části. Výsledná práce bude sloužit jako podklad pro nově vznikající katalogizační systém.

Klíčová slova

knihovní systémy, multi-knihovní systémy, webové aplikace, FRBR, OAI-PMH, MARC21, Z39.50, GWT, Ujorm

Obsah

1 Úvod

Knihovny jsou od svých počátků shromažďovateli veškerých znalostí a informací a umožňují je poskytovat ostatním lidem. Každá knihovna potřebuje nějakým způsobem katalogizovat záznamy a následně v nich vyhledávat, ať už se je to provedeno prostřednictvím informačního systému, nebo tzv. kartičkovým způsobem.

Každá knihovna v dnešní době potřebuje ILS (Integrated Library System). Existují různé druhy knihoven, a proto musí být i rozdílné informační systémy. Je důležité najít vhodný systém, který bude odpovídat instituci i jejímu rozpočtu. Mezi první kroky knihovny při výběru systém patří vzdělávání v oblastech ILS a vyhodnocení vlastních potřeb. Tyto kroky jsou nezbytné, ať už knihovna kupuje první systém, nebo přechází na jiný systém. Jen tak je možné zvolit nejvhodnější systém pro danou knihovnu (Webber, 2010).

Vývoj katalogizačních systémů se však bohužel v Evropě zpomalil, a proto je nyní většina knihovních systémů zastaralá. Je tím myšlen jak vzhled (uživatelská přívětivost), jednoduchost, tak i způsob vyhledávání. Existují však i katalogy, které používají nejmodernější technologie, ale takové katalogy většinou vznikají na vysokých školách a slouží výhradně jim, není tak prakticky možné je jakýmkoli způsobem získat pro využívání mimo tyto vysokoškolské knihovny.

Na celém světě existuje jen pár firem, které nabízí cloudových služeb pro katalogizační systémy a umožňují systém provozovat jako službu. Cloudové služby (Cloud Computing) znamená možnost sdílení software a hardware prostřednictvím sítě (nejčastěji internetu). Tyto služby jsou velmi spolehlivé, dostupné, výkonné, bezpečné a pro klienta levné.

Diplomová práce navazuje na bakalářskou práci pod názvem Systém evidence knih v obecní knihovně. Na základě této bakalářské práce firma Effectiva Solutions, s. r. o. získala obchodního partnera LANius, s. r. o., který se zabývá tvorbou katalogizačních systémů na českém i zahraničním trhu více než 20 let.

Firma LANius vyvíjí software pro knihovny od roku 1991, ale k založení firmy LANius s.r.o. došlo až v roce 1996. Nástupce katalogizačního systému LANius se začal vyvíjet v roce 1997 a na trh byl uveden v roce 1999. Tento systém se používá ve většině českých knihoven. Spolupráce s firmou Effectiva Solutions, s. r. o. vznikla oficiálně v roce 2009. Před implementací nového katalogizačního systému byl pro systém Clavius implementován nový webový katalog a Z39.50 server. Na těchto produktech byla vyzkoušena spolupráce na dálku mezi táborskou a brněnskou firmou (LANius, 2012).

2 Cíl práce

Cílem diplomové práce je návrh komplexního řešení katalogizačního systému pro knihovny aplikovaného na konkrétní společnost. Výsledky práce rovněž přímo ovlivňují vývoj nově vznikajícího systému.

Pro naplnění tohoto cíle bude provedena analýza současného trhu v zemích sousedících s Českou republikou, analýza konkurence a shrnutí požadavků knihoven. Na základě těchto informací bude navrženo komplexní řešení pro nový katalogizační systém.

Součástí práce je volba vhodného řešení, která bude spočívat v rozhodnutí mezi implementací vlastního řešení, nebo rozšířením open source řešení jako základu celého systému a volba mezi použitím open source, pořízením, nebo vývojem vybrané dílčí části katalogizačního systému.

3 Přehled aktuálního stavu

V rámci této kapitoly bude nejprve vysvětleno co jsou katalogizační systémy a následně budou popsány standardy týkající se přímo knihovních systémů. V další části kapitoly budou popsány webové aplikace a frameworky, které jsou vhodné pro tvorbu těchto webových aplikací. Na závěr budou popsány Open source licence, které umožní volbu vhodných frameworků.

3.1 Katalogizační systémy se zaměřením na knihovny

Tato část bude analyzovat kapacitu trhu a konkurence na tomto trhu. Bude popsán význam katalogizačních systémů a požadavky na tyto systémy. Dále budou popsány nejpoužívanější protokoly používající se v knihovních systémech.

3.1.1 Kapacita trhu

Pro zjištění celkové kapacity trhu je nutné nejprve rozhodnout, na jakém trhu by se nově vyvíjený systém měl nabízet. Uvedení nového produktu na trh by mělo proběhnout ve třech fázích.

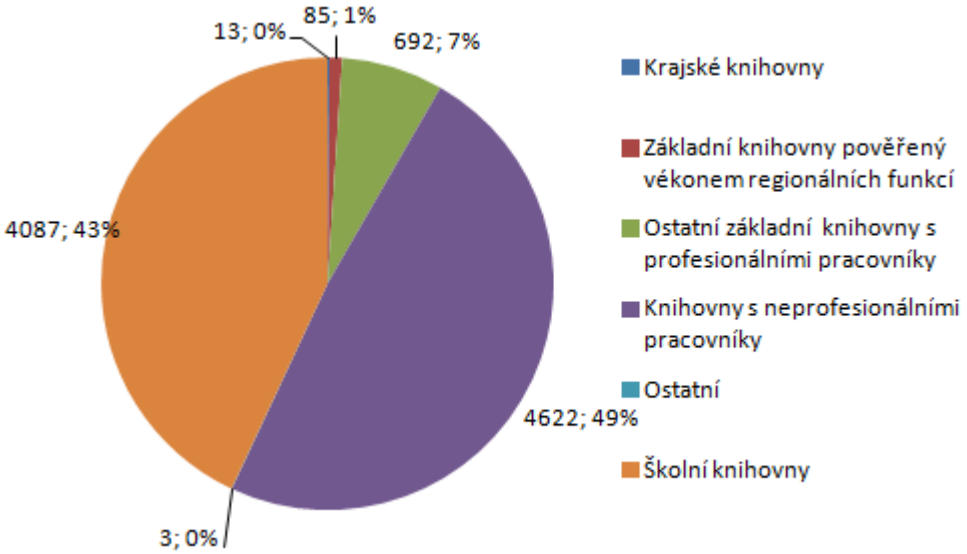
První fáze bude zaměřena pouze na Českou republiku. Je to dáno tím, že kooperující firma LANius s. r. o. je největším dodavatelem na českém trhu, má téměř 3 000 instalací. V této fázi bude produkt nabídnut stávajícím zákazníkům. Dále bude nabídnut malým obecním a školním knihovnám, které jsou pro ostatní firmy neperspektivní.

Ve druhé fázi se prodej bude zaměřovat na okolní státy, to je Slovenská republika, Polsko, Německo a Rakousko. V těchto zemích firma LANius nemá tak silné zastoupení jako v České republice, ale v Slovenské republice a Polsku již několik desítek instalací má.

Pokud se systém osvědčí na těchto trzích, bude ve třetí fázi nabízen ostatním zákazníkům bez ohledu na to, ze které země pocházejí.

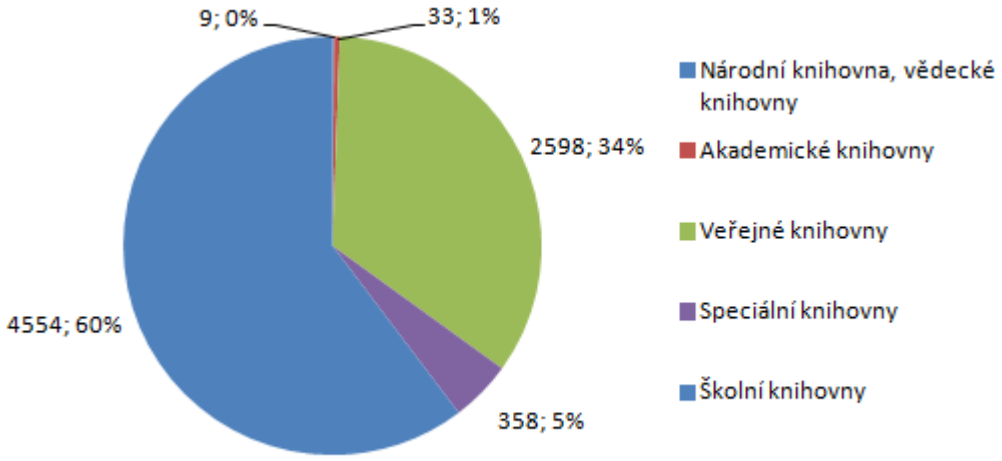
Pro získání podkladů o celkovém tržním potenciálu byly vybrány státy z první a druhé fáze. U každého státu bude zjištěn počet knihoven a jejich kategorizace. Následující grafy zobrazují počty a strukturu knihoven v jednotlivých zemích.

Česká republika



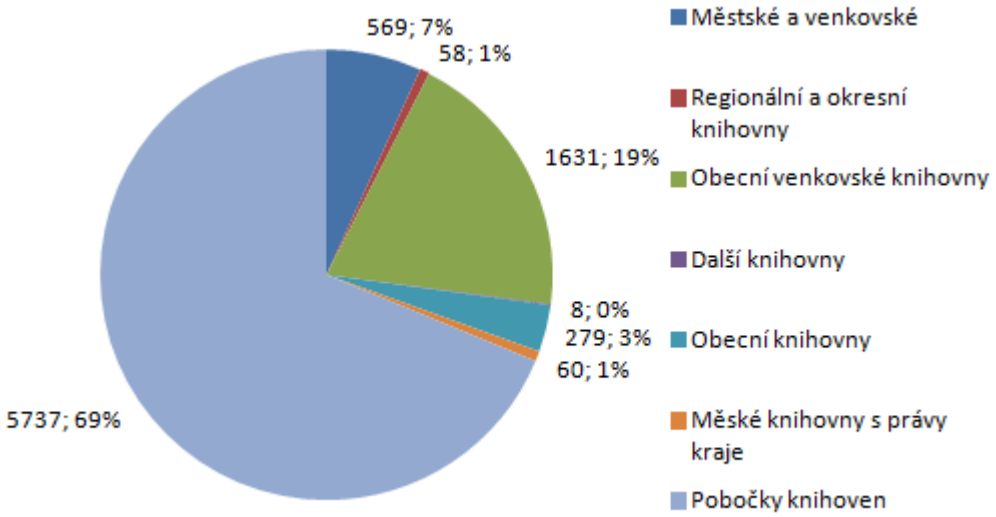
Obrázek 1: Struktura knihoven v ČR v roce 2010 (NIPOS, 2011; Čumplová, 2010)

Slovensko



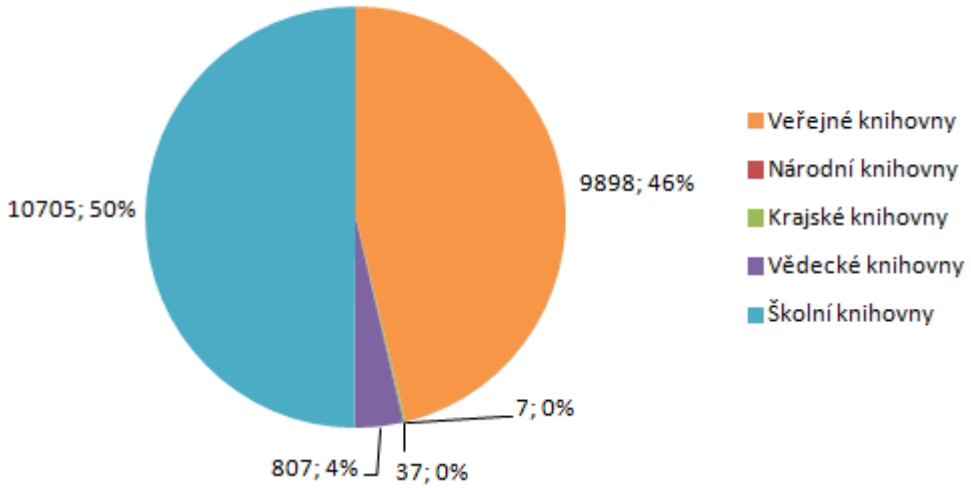
Obrázek 2: Struktura knihoven na Slovensku v roce 2009 (InfoLIB, 2011)

Polsko



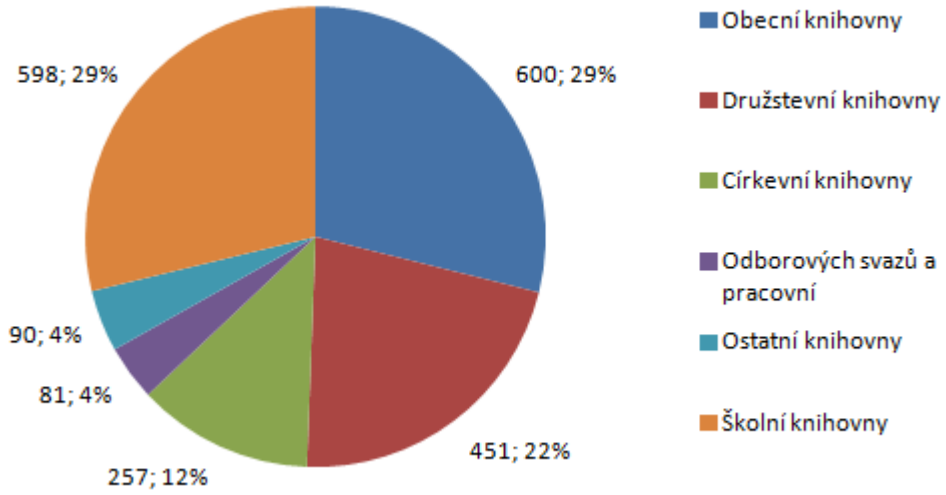
Obrázek 3: Struktura knihoven v Polsku v roce 2010 (GŁÓWNY URZĄD STATYSTYCZNY, 2010)

Německo



Obrázek 4: Struktura knihoven v Německu v roce 2010 (Schmidt, 2011)

Rakousko



Obrázek 5: Struktura knihoven v Rakousku v roce 2010 (STATISTIK AUSTRIA, 2012)

Jak vyplývá z předchozích grafů, tak jsou dvě hlavní části, na které se lze zaměřit. První z nich je oblast malých knihoven (jedná se zejména o obecní a školní knihovny), které ve valné většině nejsou automatizované, protože neexistuje žádná firma, které by se instalace systémů v těchto knihovnách vyplatily, nebo tyto knihovny nemají na nákup systému dostatek finančních prostředků. Druhou oblastí jsou velké knihovny, které se snaží jít s dobou, využívat co nejmodernější technologie, automatizovat své provozy apod. Také tyto knihovny musí šetřit své náklady a provozování klasických desktopových aplikací založených na principu klient-server je výrazně dražší než provozování webového systému jako služby.

3.1.2 Konkurence

Navrhovaný systém je zaměřen na celé spektrum knihoven od těch nejmenších až po ty největší. Systém bude navrhován tak, aby byl schopen zpracovávat velké množství záznamů a umožňoval provoz více knihoven v rámci jedné instance. To vede k tomu, že je ve výsledku jedno, jestli bude zpracovávat miliony záznamů pro jednu knihovnu, nebo v jedné běžící instanci se zaregistrují tisíce malých knihoven. Tím bude tento systém zároveň konkurencí pro katalogizační systémy zaměřené na malé i velké knihovny.

Nově vyvíjený systém má být přímým konkurentem systému Aleph, který je nejpoužívanější ve velkých knihovnách jak u nás, tak i ve světě. Tento systém má za sebou dvacetiletou historii a má přes 3 500 instalací hlavně ve velkých univerzitních knihovnách po celém světě.

Mezi další konkurenci lze zařadit systémy TinLib, KP-win, ARL, OLIB7 a další, jedná se spíše o systémy, které jsou zaměřeny na malé a střední knihovny. Nejsou zaměřeny tak vysoko jako nově navrhovaný systém a systém Aleph.

3.1.3 Katalogizační systémy

Katalogizační systémy umožňují ukládat různé množství heterogenních informací. U těchto systémů je zejména posuzován webový katalog, který je viditelný navenek a prezentuje tak celý katalogizační systém. Dalším důležitým faktorem je vyhledávání, kvalitní katalogizační systém musí rychle vyhledat požadované informace.

Jak uvádí Webber (2010), tak existují různé typy informačních systémů lišící se cenou, způsobem vlastnictví, správou systému apod. Informační systémy je možné členit na následující typy:

- Turnkey – jedná se o katalogizační systémy „na klíč“. Jeden dodavatel dodá vše potřebné pro provozování tohoto systému, jde o hardware, software, provede instalaci systému včetně instalace do lokální sítě knihovny. Dodavatelé mají vzdálený přístup a mohou tak řešit problémy, instalovat update systému a provádět různé optimalizace.
- Stand alone instalations – hardware a software je nakoupen od různých dodavatelů. Systémový administrátor nebo zaměstnanec knihovny pak sám provádí instalaci systému. Tento typ lze uplatnit ve všech typech knihoven, od těch nejmenších až po ty největší. Pro velmi malé knihovny může být instalace provedena přímo na pracovní stanici. Dodavatel ILS poskytuje pouze technickou podporu.
- Software as a service (SaaS) – nejčastěji se tento způsob používá pro webové aplikace. Na rozdíl od hostované aplikace knihovna nekupuje aplikaci, ale platí měsíční poplatky. K veškerým datům se přistupuje přes webový prohlížeč. Dodavatel zodpovídá za zabezpečení dat, jejich zálohu i dostupnost celého systému.
- Hosted system – občas dochází k záměně se SaaS. Hlavní rozdíl je v tom, že knihovna koupí licenci celého systému a dodavatel pak tento systém provozuje na vlastních serverech. Pro knihovny je to výhodné, neboť nemusí kupovat vlastní hardware a ani jej spravovat. Technická podpora dodavatele pak řeší problémy, instaluje aktualizace systému. V tomto případě je nutné mít spolehlivé připojení k internetu.
- Open source software systems – je software, který má zdrojové kódy dostupné pro individuální použití, kopírování, modifikaci i redistribuci. Toto je zásadní

rozdíl oproti systémům, které mají uzavřené (veřejně nedostupné) zdrojové kódy.

3.1.4 Multi-knihovní systém

V současnosti neexistuje žádná oficiální definice, která by definovala význam těchto slov. Pro potřeby této diplomové práce i pozdější reálné využití lze pod multi-knihovním systémem chápat následující.

Mullti-knihovní systém je vlastnost katalogizačního systému, která umožňuje katalogizování a prezentování informací více knihoven v rámci jedné instance. Knihovny mohou data katalogizovat zcela odděleně, nebo katalogizované záznamy mezi sebou sdílet. V rámci zjednodušení administrace systému je možné sdílet vybrané nastavení pro všechny nebo pouze pro vybrané knihovny.

Hlavní myšlenkou této vlastnosti systému je to, že knihovny mezi sebou mohou sdílet již vytvořené záznamy a tím dochází k minimalizaci redundantních dat v databázi. Současně to umožňuje vytvořit webové katalogy, které patří konkrétním knihovnám a lze vyhledávat pouze v jejich záznamech, ale lze vytvořit i globální webový katalog, který bude umožňovat prohledávat všechny knihovny. Pro tento společný webový katalog je důležitá vlastnost sdílení záznamů mezi knihovnami, protože tak je možné z výsledků vyhledávání vyloučit duplicity.

3.1.5 Standard MARC 21 pro strojové zpracování dat

MARC 21 je skupina katalogizačních pravidel, která umožňuje strojové zpracování bibliografických záznamů a slouží nejen pro výměnu záznamů mezi knihovnami, ale také pro ukládání katalogizovaných záznamů do databáze. Ačkoli předchůdce tohoto formátu vznikl již v 60. letech 20. století, tak neustálými revizemi a zlepšováním se formáty typu MARC staly nejrozšířenějšími (Byrne, 1998).

V rámci tohoto formátu je možné katalogizovat opravdu velké množství informací k titulům i autoritním záznamům. Informace jsou ukládány k jednotlivým podpolím, kde každé podpole má svůj pevně daný význam po celém světě, proto je možné bez problému přebírat záznamy z libovolné země. Význam každé uložené informace v podpoli může být upřesněn až dvěma indikátory. Podpole je označené čtyřpísmennou zkratkou a každé podpole patří do nějakého pole (s výjimkou polí 000-009), které seskupuje podpole podle jejich významu. Jak uvádí Národní Knihovna České Republiky (2004), tak členění polí je následující:

- identifikačních čísel - 0XX,
- kódovaných údajů - 0XX,

- popisných informací - 2XX a 3XX,
- poznámek - 5XX,
- propojovacích polí - 7XX,
- souvisejících názvů - 1XX (hlavní záhlaví), 2XX, 7XX (vedlejší záhlaví),
- předmětové analýzy - 0XX, 6XX,
- intelektuální odpovědnosti - 1XX (hlavní záhlaví), 7XX, (vedlejší záhlaví),
- zdrojových informací - 0XX.

Existují ještě národní pole, ta jsou ve tvaru 9XX, X9X nebo XX9. Tato pole si výrobci katalogizačních systémů musí registrovat a přiřadit jim patřičný význam. Při přebírání záznamů z jiných systémů je pak možné zjistit význam uložených dat. Dále je možné použít tzv. lokální pole, které mohou být v libovolné podobě např. AAA-ZZZ, těmto polím lze přiřadit význam v rámci jedné instalace systému, nebo jednoho výrobce systému.

V dnešní době je formát MARC 21 nejpoužívanějším formátem v rámci knihovních systémů po celém světě. Je to z důvodu pružného reagování tohoto standardu na nové situace, jako jsou nové typy dokumentu, neustálé úpravy standardu a zpřesňování pravidel tak, aby byly co nejvíce jednoznačné.

3.1.6 Funkční požadavky na bibliografické záznamy

Nejčastěji bývá označované jako FRBR (Functional Requirements for Bibliographic Records). Jedná se o studii funkčních požadavků na bibliografické záznamy, která vznikla v roce 1997.

Drobíková (2002) uvádí cíle studie FRBR: „Jejím záměrem je načrtnout v jasně definovaných termínech funkce bibliografického záznamu se zřetelem k různým druhům dokumentů, k různému využití a různým uživatelským potřebám. Dalším záměrem studie je doporučit základní úroveň funkcionality a základní údaje pro bibliografické záznamy vytvářené národními agenturami za účelem snížení nákladů na katalogizaci tím, že se budou vytvářet záznamy „méně než na nejvyšší úrovni popisu“, ale přesto budou vyhovovat základním uživatelským potřebám.“

FRBR se netýká pouze titulových záznamů, ale také autoritních záznamů. Studie rozšiřuje základní myšlenku knihovních systémů, že každý titul může mí N svazků. Tato studie povýšila titul na vyšší úroveň a přidělila mu větší význam. Struktura dokumentu FRBR je popsána níže (Drobíková, 2002):

- dílo – jedná se o abstraktní entitu, která seskupuje všechna možná vyjádření (bez ohledu na jazyk vydání, druh dokumentu, autora apod) určitého díla. Vzhledem k tomu, že se jedná o abstraktní entitu, je jen velmi těžké stanovovat hranice mezi tím, co je a co není nové dílo,
- exprese – forma je základní charakteristikou exprese. Při změně formy vyjádření například, pokud se z alfanumerické podoby vytvoří audiovizuální forma, pak se jedná o novou expresi,
- manifest – dříve byl označován jako dílo, umožňuje. Jde o fyzickou realizaci exprese, kde při změně fyzické normy dochází k vytvoření nové manifestace,
- jednotka – běžně bývá označován jako svazek. Je to konkrétní exemplář, který patří k určité manifestaci.

3.1.7 Z39.50 protokol pro vyhledávání a výměnu knihovních záznamů

Jedná se o protokol předávání dat zejména v knihovních systém (není to pravidlem), který vznikl již v roce 1988. Protokol byl vyvinut a používá se pro komunikaci mezi servery. Slouží jako rozhraní mezi dvěma různými informační systémy pro vyhledávání a předávání záznamů. Vyhledávat a předávat záznamy je možné jako pro titulové, tak i autoritní záznamy (Byrne, 1998).

Vyhledávání v titulových záznamech nejčastěji probíhá podle ISBN, názvu a autora. V autoritních záznamech se nejčastěji hledá přes autoritní záhlaví a uvedením typu autority. Na základě typu autority se uvádí, co je v záhlaví uvedeno – může se jednat o osobní jméno, jméno korporace apod.

Protokol Z39.50 definuje dva druhy klienta - aktivního a pasivního. Pasivní klient pouze vyhledává a stahuje záznamy ze serveru. Pro vyhledávání se používá standardizovaný dotaz, který si každý systém musí interpretovat sám podle technologie, kterou používá pro vyhledávání. Po vyhledávání se data předávají klientovi, podle toho v jakém klient požadoval data formátu (nejčastěji se používá MARC 21 a Unimarc). Naproti tomu aktivní klient umí stažené a upravené záznamy z lokální databáze odeslat zpět na server a obohatit tak záznam uložený na serveru o nově přidané informace (Byrne, 1998).

Každá knihovna může poskytovat svoje data pro potřeby ostatním knihovnám. Mezi největší poskytovatele záznamů patří Národní knihovna České republiky, která poskytuje jak titulové, tak autoritní záznamy a JIB (Jednotná informační brána) poskytující pouze titulové záznamy.

3.1.8 OAI-PMH protokol pro výměnu knihovních záznamů

Tento protokol (Open Archives Initiative Protocol for Metadata Harvesting) poskytuje interoperabilitu nezávislých aplikací využívající poskytování a stahování záznamů. Protokol se využívá zejména pro výměnu informací mezi knihovními systémy.

Protokol je založen na principu klient-server. Požadavky a odpovědi jsou předávány přes HTTP protokol ve formátu XML a v knihovním formátu MARC 21 (The Open Archives Initiative Protocol for Metadata Harvesting, 2008).

OAI-PMH se nejčastěji používá pro prohledávání a stahování záznamů z externích databází. Dalším velice častým použitím je sbírání změněných záznamů za určité období (tato doba se definuje v požadavku na data) (The Open Archives Initiative Protocol for Metadata Harvesting, 2008).

Harvester je označovaný jako sběrač dat (klient), který prohledává Providera (poskytovatele dat - server). Provider může definovat několik setů, což jsou skupiny záznamů, které je možné prohledávat. Harvester může databázi prohledávat, nebo si stahovat konkrétní záznamy (The Open Archives Initiative Protocol for Metadata Harvesting, 2008).

3.2 Obecné technologie

V rámci této části práce budou popsány technologie, které jsou vhodné pro tvorbu nového, nebo úpravu existujícího katalogizačního systému tak, aby splňoval všechny požadavky kladené na tento systém.

3.2.1 Webové aplikace

Webové aplikace, ať již hostované nebo provozované formou služby, jsou v poslední době velice populární, protože výrazným způsobem snižují náklady, odpadají starosti se zálohováním, provozem hardware a dalších činností s tím spojených.

Jak již bylo uvedeno, tak v dnešní době se značná část aplikací přesouvá na web. Vývoj nového katalogizačního systému formou desktopové aplikace by byl zřejmě krokem zpět, proto by nově navrhovaný systém měl být webovou aplikací, kterou by pak bylo možné poskytovat jako službu.

Největší výhodou webové aplikace je její dostupnost. Všechna data mohou být dostupná z kteréhokoli místa na světě jen s použitím internetu a internetového prohlížeče. U kvalitní webové aplikace by se nemělo nic instalovat (výjimkou může být nějaký plugin na speciální hardware, který usnadňuje práci se systémem, nebo je k tomu nutný). Mezi další výhody lze zařadit aktuálnost klientské části aplikace, je tak prakticky nemožné používat starou verzi a odpadají tak problémy s distribucí

klientských části aplikace jako je to u desktopových systémů založených na principu klient-server.

Webové aplikace (klientské části) lze provozovat pod libovolným operačním systémem a pro její provozování většinou postačí pouze internetový prohlížeč, takže nejsou vysoké nároky na hardware počítače. Tím klesají náklady na hardwarové a softwarové vybavení nutné pro provozování systému v rámci celé organizace.

Webové aplikace provozované jako služby mají spoustu výhod:

- levnější provoz systému,
- vysoká dostupnost systému,
- uživatel se nemusí starat o zálohování dat,
- vyšší stabilita,
- minimalizace nákladů se servisem serveru,
- minimalizace problému na straně klienta.

Další výraznou nevýhodou webových aplikací je nedodržování standardů výrobců internetových prohlížečů, jejich neustálý vývoj, automatické aktualizace prohlížečů a relativně velké množství prohlížečů, které by měla webová aplikace podporovat. Toto jsou faktory, které prodlužují dobu vývoje a mohou způsobit částečnou, nebo plnou nefunkčnost systému pro uživatele.

Při volbě vhodné technologie pro webovou aplikaci, je možné přenést část výpočetního výkonu na klienta, tím klesnou hardwarové nároky na server a tím dojde i k poklesu nákladů na koupi hardware serveru.

3.2.2 GWT - framework pro snadnou tvorbu RIA aplikací

istují různé frameworky, které usnadňují vývoj jak klientských, tak serverových částí aplikace. Technologie GWT (Google Web Toolkit) usnadňuje tvorbu klientských částí aplikace, kde umožňuje jednoduše a rychle tvořit rozsáhlé RIA aplikace, které jsou velmi podobné desktopovým aplikacím.

GWT je open source framework, které umožňuje vývoj Ajaxově založených aplikací přímo v jazyce Java. Tento framework umí zkonvertovat zdrojový kód jazyka Java přímo do JavaScriptu a HTML. Tato technologie byla uvedena poprvé v červnu 2006. Dříve bylo velmi problematické (až nemyslitelné) vytvořit RIA (rich internet application) v JavaScriptu, neboť neexistoval žádný kvalitní debuggovací nástroj. V současnosti je většina debuggovací nástrojů (jako Firebug) integrovaná přímo v prohlížeči a není tak možné debuggovat přímo zdrojový kód. GWT vyřešilo tento

problém tak, že poskytuje mechanismus, kterým je možné debuggovat klientskou část aplikace přímo ve vývojovém prostředí v jazyce Java (Gupta, 2008).

Celé GWT je založeno na tom, že programuje v jazyce Java a kompilátor zkompiluje Java kód do JavaScriptu a HTML pro každé jádro internetového prohlížeče zvlášť. Výhodou pak je, že se uživateli do internetového prohlížeče stáhne balík JavaScriptů, které jsou pak spuštěny na klientovi. Oproti např. JSF server nemusí generovat žádné soubory, pouze odešle klientovi aplikaci.

Tato technologie má odstínit vývojáře od ladění chyb pro všechny prohlížeče a umožňuje mu plně se soustředit na vývoj aplikace. Tím má vývojář více času na programování a stává se tak efektivnější.

Podle Dwyer (2008) má technologie GWT následující vlastnosti:

- Podporuje škálovatelnost – pokud bylo GWT vyvinuto pro jeden důvod, tak je to právě škálovatelnost. Při vývoji Gmailu a Google Maps založených na JavaScriptu začal Google vyvíjet vlastní nástroj, kdy zaměstnal nejlepší vývojáře JavaScriptu na světě aby vyvinuli tuto novou technologii.
- Snadno se refactoruje – v JavaScriptu se obecně rychle vytvářejí prototypy, ale refactorování těchto zdrojových kódů nepatří k silným stránkám. Jelikož vývoj probíhá přímo v jazyce Java, je možné využít kvalitní IDE, tvořit vlastní třídy apod. je pak možné využít všech nástrojů jako u klasické desktopové aplikace.
- Podobnost – GWT velmi připomíná Swingové komponenty, nebo ostatní standardní grafické rozhraní. Tato technologie je založena na asynchronním volání metod (RPC – Remote Procedure Call), což může ze začátku programátorům působit problémy, než si na tuto vlastnost vzniknou.
- Rychlost – na webu je rychlost to nejdůležitější. Kompilátor mezi kompilací Java zdrojového kódu do JavaScriptu provádí velké množství operací, které způsobují to, že výsledný kód je v daném prohlížeči co nejrychlejší. Současně s tím provádí obfuskaci zdrojového kódu tak, že zkracuje názvy proměnných a metod tak, aby výsledná aplikace byla co nejmenší, dále se provádí komprese, aby výsledné stažení aplikace bylo co nejrychlejší.
- Spousta nástrojů – pro vývoj lze použít spoustu nástrojů jako jsou Netbeans, Eclipse, IntelliJ, IDEA apod. Samozřejmostí je syntaktická kontrola zdrojových kódů a automatická kompilace na pozadí. Podpora PMD což umožňuje procházet zdrojové kódy, hledat možné chyby, mrtvý, neoptimální, duplicitní a zbytečně složitý kód.
- JavaScriptové knihovny – je možné importovat libovolnou JavaScriptovou knihovnu, ke které se přistupuje přes rozhraní JSNI (JavaScript Native Interface).

- Komunita – o mnohých technologiích se tvrdí, že mají kvalitní a aktivní komunitu uživatelů. Komunita uživatelů GWT je opravdu velká, tito uživatelé mezi sebou sdílí znalosti formou různých prezentací, konferencí apod. Také si mezi sebou pomáhají řešit různé problémy. Na základě těchto faktů lze jen předpokládat, že se tato technologie bude dále rozvíjet.

3.2.3 Objektové mapování relačních databází

Pro objektové mapování relačních databází byla nalezena technologie Ujorm. Jedná se o novou technologii využívající JDBC (Java Database Connectivity) a umožňuje snadný přístup k datům, zároveň je jeho konfigurace velmi jednoduchá, neboť tento framework nenabízí takové rozsáhlé možnosti jako konkureční framework Hibernate nebo iBatis. Tato technologie se velmi hodí pro aplikace používající technologii GWT.

Podle Ponce (2009) tato technologie vznikla původně jako podpora architektury UJO (Unified Java Object), která je alternativou architektury POJO (Plain Old Java Object).

Ujorm je nízkoúrovňový ORM framework, který se používá u Java aplikací pro mapování relačních databází. Tento framework se velmi hodí k požadované technologie GWT, ale zatím nebyl použit v žádné komerční aplikaci.

Oproti technologii Hiberante, která je jedna z nejpoužívanějších u Java aplikací, tento framework nenabízí tolik možností. Nabízí však vše potřebné, co je nutné k použití v každé aplikaci. Proto je také podstatně snadnější na použití i databázové dotazy jsou rychlejší než u frameworku Hibernate. Rychlejší je z několika důvodů, jednak proto, že nedochází k žádnému parsování SQL dotazu a také proto, že Ujorm nepoužívá Java reflexi (Ponec, 2009).

Ujorm podporuje téměř všechny typy databází – Oracle, PostgreSql, MSSQL, MySQL, Derby a další.

Ponec (2009) uvádí velmi užitečné funkce této technologie:

- správa session a transakcí ve frameworku Spring,
- generování primárních klíčů – o přidělování se nestará databáze, ale přímo Ujorm, v některých případech potřebujeme (je vhodné) znát ID ukládaného objektu dříve, než jej databáze skutečně uloží,
- automatické pojmenování primárních klíčů, cizích klíčů, indexů a constraintů. Toto je velmi důležitá vlastnost, která je velmi důležitá při vývoji produktů. Pojmenování se nenechává na databázi, která generuje náhodné řetězce a u produktů používající různé databáze se indexy mohou jmenovat jinak a není možné jednoduchým způsobem vytváření migrací,

- automatický překlad klientských objektů na serverové a zpět – pokud vyvíjený systém používá dva typy objektů (jedny klientské a druhé serverové), pak se běžně pro každou dvojici objektů musí vytvořit vlastní translator a při přidání, nebo odebrání atributu v daném objektu se ještě musí upravit translator. Jelikož Ujorm vychází z technologie Ujo, odpadá nutnost tvorby těchto translatorů a na předávní hodnot lze využít pouze jeden translator, který je implementovaný přímo v Ujormu,
- lazy loading – umožňuje načítání objektů v době, kdy jsou skutečně potřeba. Pokud máme například objekt Auto, který má objekt Motor, tak dokud se na objekt Motor nezeptáme, tak se objekt Motor z databáze nenačte,
- výkonové optimalizace – jelikož běžné načítání i lazy loading může v případě načítání kolekcí vést k neúměrnému zatěžování databáze, tak Ujorm nabízí optimalizace, které minimalizují počty dotazů. Pokud načítáme například kolekci Aut a ke každému automobilu potřebujeme znát jeho objekt Motor, tak v tomto případě se data načtou pouze dvěma dotazy,
- typově kontrolované parametry – automaticky podporuje tzv. „bezpečné parametry“, které znemožňují použití SQL injection,
- session cache – tato funkce má omezit zbytečné volání SQL dotazů. Tato cache má platnost v rámci databázové transakce, po zavolání rollback, nebo commit se automaticky uvolní. Volání databázových procedur a funkcí – pro velké systémy je vhodnější spíše používání databázových procedur než aplikační vrstvy v systému, neboť procedury jsou rychlejší a bezpečnější.

3.2.4 LDAP autentizace

Lightweight Diretory Access Protocol (LDAP) jedná se o protokol, který se používá pro přístup k datům na adresářovém serveru. Existuje mnoho použití pro tuto službu, tímto způsobem lze organizovat různé typy informací.

Jedná se o jednu z nejpoužívanějších forem autentizace, a proto by nově navrhovaný systém měl podporovat tento způsob autentizace.

Struktura LDAP je založena na adresářích, které se skládají ze záznamů. Informace jsou přiřazovány k atributům, což jsou části záznamů. Atributům lze nastavovat povinnost, datový typ, hashování a omezení. V jednom adresáři může být uloženo více typů záznamů například údaje o osobách, produktech a podobně. LDAP ještě obsahuje kontejnery, které umožňují lépe organizovat vztahy mezi záznamy. Nejčastěji se kontejnery používají pro tvorbu organizačních jednotek označované ou. Informace jsou ukládány v adresářové struktuře, ale jejich použití se netýká pouze

lidí, ale všeobecně libovolných objektů reálného světa. Všechny záznamy musí obsahovat dn (distinguished name), které jsou unikátní a je tak možné jednoznačně identifikovat každý uložený záznam (Arkills, 2009).

Služba je založena na principu klient-server a je provozována pod protokolem TCP/IP (je možné ji provozovat přes SSL a TLS).

LDAP se nejčastěji používá pro autentizaci uživatelů, umožňuje tak používat jedny přihlašovací údaje pro programy používající LDAP jako autentizační službu. Tento protokol toho nabízí mnohem víc, umožňuje uchovávání informací o uživatelích, nastavení jednotlivých aplikací apod. Velmi často se používá pro uchovávání nastavení zařízení, které musí mít informace uloženy a spravovány z jednoho místa (Arkills, 2009).

3.3 Licence otevřeného software

Pokud se při vývoji systému používají frameworky třetích stran, pak je důležité mít jasno v licencích, pod kterými jsou tyto frameworky vydány. U rozsáhlých aplikací je použití frameworků třetích stran prakticky nevyhnutelné, protože toto použití výrazným způsobem šetří čas a náklady na vývoj nového systému. V této kapitole budou popsány nejpoužívanější typy licencí otevřeného software (open source license).

Open source software má dostupné veřejně a zdarma všechny svoje zdrojové kódy, ale pozor, není open source licence jako open source licence. Při volbě vhodného projektu (frameworku) je třeba dbát na licenci, pod kterou je tento projekt (framework) vydán. Existuje celá řada licencí, které umožňují volné užití projektu. Některé licence však umožňují volné užití pouze k nekomerčním účelům, ale existují i licence, které lze použít i pro komerční účely.

Mezi nejpoužívanější licence patří GNU GPL, GNU LGPL a Apache licence, Version 2.0.

3.3.1 GNU General Public License

Jedná se o jednu z nejpoužívanějších licencí. Frameworky vydané pod touto licencí je možné používat i pro komerční využít.

Díla vydaná pod tou to licencí je možné upravovat a rozšiřovat, ale výsledné dílo musí být vydáno pod licencí GNU GPL. V případě vývoje vlastní aplikace, která používá dílo vydané pod touto licencí, pak i celé výsledné dílo musí být vydáno pod licencí GNU GPL (GNU, 2007).

3.3.2 GNU Lesser General Public License

Tato licence byla založena jako kompromisní řešení mezi licencemi GNU GPL a BSD licence. Za hlavní rozdíl lze považovat, že při použití frameworku vydaného pod touto licencí nemusí být výsledný program vydaný pod licencí GNU LGPL. Jak tato licence uvádí, tak dílo používající tento framework a není označeno jako odvozené dílo. Pokud se jedná o modifikaci díla vydaného pod touto licencí, pak modifikace tohoto díla musí být vydána pod licencí GNU LGPL (GNU, 2007).

Použití této licence je vhodné pro komerční aplikace, protože výsledné dílo může být vydáno pod libovolnou licencí a není nutné zveřejňovat zdrojové kódy celého díla.

3.3.3 Apache licence, Version 2.0

Tato licence je jedna z nejsvobodnějších licencí. Frameworky vydané pod touto licencí je možné použít i komerčních aplikacích a výsledný projekt je možné vydat pod libovolnou licencí, nebo si ponechat zdrojové kódy pouze pro sebe. Původní zdrojové kódy musí obsahovat informace o tom, pod jakou licencí byly vydány. (The Apache Software Foundation, 2004).

Stejně jako u licence GNU LGPL se je tato licence vhodná pro komerční aplikace.

4 Použité metody

V rámci této diplomové práce jsou použity metody pro podporu rozhodování a objektové modelování systému.

4.1 Metody pro podporu rozhodování

Tato část popisuje metody, které slouží pro podporu rozhodovacích procesů při volbě vhodných projektů nebo frameworků. Špatné rozhnodnutí může mít zásadní vliv na produkt jako celek, proto musí být výsledné rozhodnutí podloženo fakty.

4.1.1 Vícekriteriální rozhodování

Každé rozhodování je závislé na kritériích, které jsme si zvolili, nebo které byly zvoleny někým jiným. Pokud máme pouze jedno kritérium, je rozhodování jednoduché. Jakmile máme kritérií více, je rozhodování složitější a celý proces rozhodování trvá déle (Tichý, 2006).

Tato metoda byla vybrána proto, že umožňuje najít optimální řešení pro danou situaci. Metoda je použita pro volbu vhodného indexačního a vyhledávacího frameworku, protože na tento framework je kladeno mnoho různorodých požadavků, kde některé jsou velmi důležité a některé naopak málo důležité.

V případě vícekriteriálního rozhodování je velmi důležitá správná volba kritérií a to jak z hlediska kvantity, tak i kvality. Problémy nejčastěji vznikají při větším počtu kritérií a rozhodovatelů. Pokud rozhoduje skupina osob, musí dojít k jejich konsenzu, který může být definován jednomyslným souhlasem, souhlasem kvalifikované většiny, prosté většiny nebo jinak. Vícekriteriální rozhodování musí být správně definováno a řízeno, jinak se zvyšuje pravděpodobnost výskytu problémů. Mezi největší problémy lze zařadit výskyt protichůdných kritérií (Tichý, 2006).

V rámci této práce bude vícekriteriální rozhodování použito pro výběr vhodného indexačního frameworku, jako dílčí části katalogizačního systému.

Výsledkem těchto rozhodování bývá určování pořadí jednotlivých variant, nebo podklad pro přijetí/zamítnutí dané varianty. Je důležité mít na paměti, že pořadí může být z části náhodné a že to může mít vliv na závěrečné rozhodnutí. Je velmi dobré rozlišit kritéria podle jejich důležitosti, k těmto kritériím jsou přiřazovány váhy, podle významu každého kritéria. Občas bývá vhodné zakázat změnu váhy kritérií v průběhu rozhodovacího procesu, protože lze tak zásadním způsobem ovlivnit výsledek rozhodování (Tichý, 2006).

Metod používajících se při vícekriteriálním rozhodování je celá řada, mezi základní lze zařadit následující metody (Brožová, Šubrt, Houška, 2003):

- Entropická metoda – tato metoda je založena na hodnocení jednotlivých variant, čím je hodnocení kritéria podobnější, tím nižší váhu toho kritérium má, pokud jsou všechna hodnocení stejné, kritérium není důležité a lze jej vynechat. Čím vyšší je rozdíl v hodnocení kritéria, tím větší váhu toto kritérium má. Pro určování vah se proto používá míra entropie.
- Metoda pořadí – při použití této metody hodnotí jednotlivá kritéria skupina expertů, kdy každý člen určuje pořadí daného kritéria od nejdůležitějšího po nejméně důležité. Váhy se určují tak, že se u každého kritéria sečtou hodnocení od všech členů a vydělí se celkovým počtem bodů.
- Bodovací metoda – v rámci této metody je nejprve nutné stanovit bodovací stupnici. Každému kritériu se přiřadí počet bodů ze zvolené stupnice, čím vyšší počet bodů je přidělen, tím je kritérium důležitější. Výpočet vah k jednotlivým kritériím se provádí stejně jako u metody pořadí.
- Saatyho metoda – tato metoda se používá v případě, že hodnotitelem je pouze jedna osoba. Metoda je založena na kvantitativním párovém porovnání jednotlivých kritérií. Používá se devítibodová stupnice, kde hodnotu jedna mají rovnocenná kritéria a hodnotu devět mají kritéria, kde jedno z nich je absolutně preferované před druhým. Pro stanovení vah se nejčastěji používá výpočet jako normalizovaného geometrického průměru řádků v Saatyho matici.

4.1.2 Analýza prostředí

„Cílem SWOT analýzy je identifikovat to, do jaké míry jsou současná strategie firmy a její specifická silná a slabá místa relevantní a schopná vyrovnat se změnami, které nastávají v prostředí.“ (Jakubíková, 2008).

Tato analýza bývá často velmi užitečná při rekapitulaci více analýz, slouží také při identifikaci využití unikátních zdrojů nebo klíčových kompetencí firmy. Tato analýza má také své nevýhody. Mezi hlavní nevýhodou je to, že je příliš statická a také subjektivní. Pro tvorbu strategický dokumentů tato metoda nemá velký užitek (Jakubíková, 2008).

SWOT analýza byla vybrána proto, že umožňuje přehledně srovnat zejména silné a slabé stránky. Analýza nemusí být provedena pouze na firmu jako celek, může být použita také jako prostředek analýzy produktu. Tato analýza slouží jako podklad pro rozhodování při volbě open source projektu jako základ celé aplikace.

Analýza se stává z původně dvou analýz vnitřního a vnějšího prostředí. Nejprve se doporučuje začít analýzou vnějšího prostředí a to jak makroprostředí, tak i mikroprostředí. Po analýze vnějšího prostředí se provádí analýza vnitřního prostředí firmy (Jakubíková, 2008).

Při použití této metody dochází k analýze vnitřních a vnějších faktorů. Do vnitřních faktorů se řadí silné a slabé stránky a do vnějších faktorů se řadí příležitosti a hrozby. Tyto faktory jsou často členěny do tzv. SWOT matice (Jakubíková, 2008):

- Silné stránky (strengths) – jedná se o faktory, které přináší určité výhody pro zákazníky, ale i pro firmu samu. Často tyto faktory ukazují na konkurenční výhody v porovná s ostatními firmami.
- Slabé stránky (weaknesses) – označují problematická místa ve firmě, nebo místa, ve kterých jsou ostatní firmy úspěšnější. Tyto faktory jsou velmi důležité a je u nich prostor pro zlepšení.
- Příležitosti (opportunities) – uvádějí faktory, díky kterým je možné zvýšit poptávku, umožňuje vyšší uspokojení zákazníků nebo mohou přinést úspěch celé firmě.
- Hrozby (threats) – uvádí možnosti pomocí kterých může dojít ke snížení poptávky nebo způsobit nespokojenost zákazníků.

Tato analýza se často používá ke shrnutí ostatních analýz, může se také používat k identifikaci možnosti použití unikátních zdrojů nebo klíčových kompetencí firmy. Tyto analýzy jsou často velmi subjektivní a to může být velkou nevýhodou (Jakubíková, 2008).

4.2 Objektové modelování systému

V této části jsou popsány diagramy, které budou použité při návrhu katalogizačního systému. Popsané diagramy umožní namodelovat systém podle aktuálních požadavků.

4.2.1 Případy užití

Požadavky na informační systém se rozlišují jako funkční a nefunkční, případy užití (označované jako use case) zachycují právě funkční požadavky na systém. Případy užití označované jako use case popisují interakce mezi uživatelem a systémem, a dávají nám informace o tom, jak je systém používaný (Fowler, 2009).

Tyto diagramy se používají již při návrhu modelu, které slouží jako podklad pro specifikaci požadavků na informační systém. Nejčastěji se používají pro komunikaci se zákazníkem, specifikaci požadavků. Diagramy případů užití byly zvoleny právě proto, že umožní graficky zobrazit informace o tom, jak je systém používaný a tato forma je srozumitelná pro všechny zúčastněné strany. Tyto diagramy usnad-

ňují řízení projektu pro projektové manažery, pro navrhování GUI i komunikaci se zákazníkem.

Use case diagramy popisují vztahy mezi aktory a use casy. Jak uvádí Fowler (2009), tak posloupnost kroků popisující interakci mezi uživatelem a systém je označována jako scénář. Aktor (reprezentován panáčkem) je libovolná osoba nebo externí systém, který přijímá hodnoty z use case. Use case (reprezentovaný oválem s krátkým popisem) popisuje případ užití, které provádí aktor nebo s ním interaguje (Leander, 2000).

Diagramy zobrazují aktory a případy užití, ve kterých se vyskytují. Z diagramu nemůžeme určit pořadí, ve kterém by měly být případy užití použity nebo pořadí jednotlivých aktivit, pro tyto případy je nutné použít diagram aktivit. (Podeswa, 2009).

Diagramy pomáhají vizualizovat a organizovat případy užití, ale neposkytují žádné informace kromě názvů případů užití. Use case diagramy jsou nejjednodušší formou pomocí které je možné vyjádřit, jak uživatelé interagují s počítači. Vysvětlují souvislosti mezi interakcemi, například proč osoba provádí tento úkol a jaké kroky existují od začátku do konce. Také popisuje výjimky podmínek a co se stane, když k výjimce dojde. Je napsaný v business jazyce, který je srozumitelný jak pro business uživatele, tak i pro programátory (Leander, 2000).

4.2.2 Diagram tříd

Diagram tříd (class diagram) představuje statický pohled na modelovaný systém, protože není možné vytvořit interakci mezi třídami. Základem tohoto diagramu je třída, která je brána jako abstrakce objektů se stejnými vlastnostmi, chováním a vztahy k ostatním objektům. Ty třídy, které mohou vytvářet instance se nazývají konkrétní třídy a ty třídy, které instance vytvářet nemohou se označují jako abstraktní třídy (Bruckner, 2012).

Tyto diagramy umožňují modelovat hierarchii tříd a jsou obecnější než ER (entity – relationship) diagramy, protože umožňují zobrazovat více informací, které jsou potřebné pro analýzu i návrh systému. Proto byly diagramy tříd zvoleny pro návrh systému.

Diagram tříd zobrazuje každý objekt nebo definici třídy ve vizuální formě s druhy různými šipek, které zobrazují vztahy mezi objekty. Každá třída je reprezentována boxem, který je rozdělen do tří částí ve kterých je uveden název třídy, seznam atributů a seznam metod (Leander, 2000).

Mezi třídami mohou vznikat různé druhy vazeb, asociace, kompozice, agregace a generalizace (Leander, 2000).

- Asociace je logická vazba mezi dvěma třídami často přes klíčovou hodnotu, která odkazuje na jiný objekt. Asociace poskytuje navigaci z jednoho objektu na jiný. Může se jednat o vazbu 1:1 nebo 1:N. Vazba je zobrazena čarou a šipkou.
- Kompozice je cesta, jak zobrazit třídu jako atribut jiné třídy. Tato vazba je zobrazena čarou s vyplněným kosočtvercem.
- Agregace je něco mezi asociací a kompozicí, jedná se o silnější vazbu než asociace, ale může být implementována stejně. Vazba je zobrazena čarou s prázdným kosočtvercem.
- Generalizace je proces rozšíření jedné třídy do nové, mnohem specifičtější třídy. Všechny public a protected atributy a metody nadřazené třídy jsou dostupné v podřízené třídě. V podřízené metodě je možné přidat vlastní atributy a metody nebo je možné upravit metody nadřízené třídy. Vazba je zobrazena čarou s prázdným trojúhelníkem.

5 Metodika práce

Na základě analýzy aktuálního stavu na poli katalogizačních systémů a aktuálních požadavků (funkčních i nefunkčních) je v rámci hlavní části práce navržena architektura nového systému.

Po volbě architektury aplikace dojde k vytvoření nového návrhu katalogizačního systému, ať už je zvolen open source projekt, nebo je zvolena implementace celého systému od začátku.

Další část vlastní práce se věnuje volbě dílčí části systému u které se rozhoduje mezi použitím open source, koupí nebo implementací vlastního frameworku. Rozhodnutí je provedeno zejména na základě vybrané metody vícekriteriálního rozhodování. Nejprve se zvolí jednotlivá kritéria a každému z nich se podle důležitosti přiřadí jeho váha. Každá varianta u každého kritéria je ohodnocena příslušným počtem bodů. Výsledek tohoto vícekriteriálního rozhodování slouží jako podklad pro závěrečnou volbu.

V diskuzi je zhodnocen aktuální stav vývoje katalogizačního systému. Dále je výsledný návrh systému srovnán oproti původním požadavkům na katalogizační systém. Jsou také diskutovány otázky, které postupně vyvstávaly při tvorbě této práce. V rámci diskuze je provedeno srovnání odhadů na implementaci celého systému s reálnou dobou implementace. Závěr práce se zhodnotí výsledky, které byly dosaženy ve vlastní práci.

6 Vlastní práce

6.1 Koncepční návrh aplikace

Volba vhodné koncepce pro základ celé aplikace je jedním z nejdůležitějších rozhodnutí, protože na základě této volby bude implementován, provozován a v budoucnu rozšiřován celý systém. Náročnost hledání chyb, vývoje nových funkcí, případně úprava stávajících funkcí bude záležet právě na tomto rozhodnutí.

V tomto případě přichází v úvahu využití již existujícího open source řešení, které by se dalo převzít a dále rozvíjet, nebo vydat se cestou implementace celého systému od začátku. Nejprve je však nutné zjistit aktuální požadavky na katalogizační systém.

6.1.1 Požadavky na systém

Aby bylo možné se správně rozhodnout při volbě základu celého systému, volbě jednotlivých dílčích částí a návrhu celého systému je nutné provést kvalitní a podrobný sběr požadavků.

Požadavky jsou rozděleny na funkční a nefunkční. Pro provoz systému musí být splněny minimálně následující požadavky.

Nefunkční požadavky:

- Obecný katalogizační systém – nově vznikající systém bude ctít knihovní pravidla, ale nemá být čistě knihovním systémem, musí umožňovat ukládání různorodých informací.
- MARC 21 – většina knihovních katalogizačních systémů používá právě tuto normu. Jelikož se jedná v současnosti o nejpoužívanější formát, proto systém bude podporovat ukládání a výměnu záznamů v tomto formátu.
- FRBR – tato studie umožňuje lepší členění katalogizovaných záznamů a na základě tohoto členění je uživateli zpříjemněno vyhledávání. Uživatel si tak může jednoduše a rychle najít veškeré katalogizované informace související s vyhledaným dílem.
- Provoz na unixových operačních systémech – systém musí být provozovatelný pod unixovými operačními systémy. Důvodem těchto systému je výborná stabilita, jednoduchá správa a rychlost souborového systému. Výhodou je možnost provozu pod operačním systémem Windows Server 2003 a vyšší.
- Moderní, uživatelsky přívětivý IS – jelikož nově vznikající systém má být webová aplikace a uživatelé jsou zvyklí na pohodlí desktopových aplikací, pak uživatel-

ské rozhraní musí co nejvíce přiblížit desktopové aplikaci a přitom uživatel by neměl být žádným zásadním způsobem omezen.

- Moderní webový katalog – vzhled většiny současných webových katalogů vypadá, jako by se jejich vývoj pozastavil před deseti lety. Uživatelé si postupně zvykají na jednoduché hledání pomocí jednoho vyhledávacího okénka a také na přehledné zobrazení. Webový katalog by měl splňovat požadavky na Web 2.0.
- Podpora ukládání velkého množství dat – jelikož se jedná o multi-knihovní systém, systém musí podporovat ukládání dat s miliony titulů. Systém bude podporovat následující tři relační databáze:
  - Oracle 10g a novější,
  - MSSQL 2008 a novější,
  - MySQL 5.1 a novější.

  Tyto databáze byly vybrány proto, že velké univerzitní knihovny používají nejčastěji Oracle databázi a větší knihovny používají databáze MSSQL. Použití databáze MySQL je z toho důvodu, aby v případě speciální instalace si knihovna nemusela kupovat ještě licenci na databázi. Verze k jednotlivým databázím byly zvoleny proto, že není vhodné podporovat všechny verze jednotlivých databází.
- Podpora malých knihoven - existuje velké množství malých obecních a školních knihoven, které nemají žádný katalogizační systém, protože na nákup nového systému nemají dostatek finančních prostředků. Na této části trhu je velký prostor pro webový, multi-knihovní systém poskytovaný jako hostovaná služba (SaaS).
- Možnost provozu malých i velkých knihoven v rámci jedné instance – malé knihovny mají určité rozdíly oproti velkým knihovnám a proto se k těmto knihovnám musí přistupovat rozdílně. Největším rozdílem je, že u malých knihoven nepracují zkušení knihovníci.
- Multi-knihovní systém – toto je jeden ze zásadních funkčních požadavků. V jedné instanci systému budou existovat virtuální knihovny, které mezi sebou budou moct sdílet data. Systém tak může být provozován formou služby, kde se jednotlivé knihovny zaregistrují a systém budou moct ihned využívat.
- Webová aplikace – jedná se o další zásadní požadavek. V současné době se velké množství aplikací přesouvá na internet, dokonce vznikají i internetové operační systémy.

Funkční požadavky:

- Zvýšení automatizace – Čárové kódy se knihovnách již běžně používají, je proto nutné posunout se krok vpřed a podporovat nové technologie pro hromadné zpracování dat – RFID. Dalším požadavkem pro zvýšení automatizace jsou samoobslužné pulty. Jsou to pulty, u kterých se nevyskytují zaměstnanci knihovny, ale půjčování a vracení knih provádí sami čtenáři.
- Podpora chytrých telefonů – Chytré telefony mohou být v knihovnách používány z několika důvodů:
  - vyhledávání záznamů ve fondu,
  - navigace po knihovně – možnost čtenáře navigovat přímo k regálu s požadovanou knihou,
  - revize – chytré telefony již běžně obsahují čtečku čárových kódů a některé podporují i čtení RFID kódů. Tyto telefony pak výrazně zjednodušují inventarizaci.
- Možnost LDAP autentizace – jelikož nově vznikající systém má být obecný katalogizační systém, pak lze předpokládat, že některé organizace budou požadovat nasazení na jejich vlastních serverech a budou požadovat autentizaci prostřednictvím LDAP serveru.
- Podpora více jazyků – do systému se bude povolena registrace kterákoli instituce z libovolné země. Systém musí umožňovat dynamické přidávání jazyků. Všechny překlady by proto měly být ukládány v kódování UTF8.
- Přístup k digitálním dokumentům – podpora digitalizace textových dokumentů, ochrana těchto dokumentů.
- Webový katalog:
  - Vyhledávání – stejně jako u vlastního katalogizačního systému, tak i webový katalog musí podporovat vyhledávání pomocí jednoho okénka, řazení podle relevance atd.
  - Napojení katalogu na sociální sítě – samozřejmostí je napojení na sociální sítě jako je Google+, Facebook, Twitter a spousta dalších. Napojení na sociální sítě umožňuje sdílení oblíbených knih ve svém profilu, doporučování svým přátelům apod.

- Sdílení hodnocení mezi uživateli – v rámci všech webových katalogů, které nemusí být všechny v rámci jedné instance se bude sdílet hodnocení mezi čtenáři. Budou existovat dva typy hodnocení:
  * hvězdičkami – každý titul může být hodnocen hvězdičkami. Oblíbenost titulu by měla být částečně zohledněna při vyhledávání podle relevance – oblíbenější tituly by se měly zobrazovat na vyšších pozicích,
  * diskuze – ke každému titulu budou čtenáři moct přidat slovní hodnocení (komentář). V rámci podpory slovního bude možné ohodnotit každý komentář stylem líbí/nelíbí. Bude tak možné zobrazovat nejvíce a nejméně oblíbené komentáře u každého titulu.
- Napojení na externí zdroje – webový katalog by měl podporovat napojení na externí zdroje jako je například Naxos. Většinou je možné zdroje používat pouze z intranetu knihovny, jsou však požadavky na to, aby tyto služby byly umožněné čtenářům z jejich domova po úspěšné autentizaci do webového katalogu.
- Podpora slepeckých zařízení pro nevidomé – jelikož knihovny půjčují nevidomým lidem namluvené audio nahrávky knih, tak by měly podporovat i možnost přístupu nevidomých na internetové stránky a umožnit jim vyhledávání v katalogu. Knihovny jsou většinou zřizované městem, proto by měly umožňovat přístup všem lidem bez ohledu na jejich postižení.

6.1.2 Open source řešení

Nejlepší variantou často bývá využít již existující projekt a rozšířit jej podle svých požadavků, neboť jsou v tomto projektu již implementovány základní funkce.

Pro volbu základu aplikace byl preferován open source projekt (vydaný pod licencí LGPL nebo Apache licence, Version 2.0), který by bylo možné využít i pro komerční využití. Žádný takový projekt, který by měl potenciál stát se úspěšným produktem nebyl nalezen.

Bylo tak nutné zaměřit se na open source projekty, které jsou volné pouze k nekomerčnímu použití (zejména licence GPL) a pokusit se odkoupit licenci pro komerční prodej tohoto open source projektu, nebo změnit business plán a místo prodeje SW licence by se musely platit poplatky za instalaci, hostování aplikace apod. Bylo nalezeno několik projektů vytvořených pod licencí GPL, které by připadaly v úvahu. Byly vybrány následující čtyři potenciální projekty, které by moly být použity jako základ nově vznikajícího systému.

Byly nalazeny následující open source projekty: Greenstone, NewGenLib, Koha a Evergreen.

Greenstone

Greenstone má více než 4 500 zákazníků, jde spíše o menší knihovny. Jedná se o desktopový systém založený na jazyce Java a Perl. Mezi jeho výhody lze zařadit například podpory ve streamování videa a možnost tvorby překladů pro libovolné jazyky (pouze ve zdrojových kódech).

Tabulka 1: SWOT tabulka pro projekt Greenstone

Vnitřní prostředí	Silné stránky • Základ v Jazyce Java • Podpora OS Windows, Linux a Mac • Možnost snadného přeložení do více jazyků • Videostreaming server		Slabé stránky • Desktopová aplikace • Podpora pouze MySQL databáze • Problematická instalace klienta
Vnější prostředí	Příležitosti • Vznik nových dotačních programů • Lepší využití tržního potenciálu • Změna segmentace trhu • Využití lepších technologií		Hrozby • Nutná změna business plánu kvůli GPL licenci • Stávající konkurence na trhu • Zrušení stávajících dotačních programů • Využití zdrojových kódů konkurencí

NewGenLib

Systém se používá ve 30 zemích světa, kde má více než 2 700 instalací. Tento systém je plně webový, je programovaný opět v jazyce Java a využívá technologií appletů z Java Web Start. Nevýhodou je, že pro používání klienta musí být na klientském počítači instalované JRE. Používá databázi PostgreSQL, ale z důvodu použití technologie Hibernate by nemělo být velkým problémem popdorovat i jiné databáze. Pro vyhledávání používá technologii Solr a pro prezentační část webového katalogu technologii JSP (JavaServer Pages).

Tabulka 2: SWOT tabulka pro projekt NewGenLib

Vnitřní prostředí	Silné stránky • Základ napsaný v jazyce Java • Nový systém - podpora aktuálních požadavků • Podpora škálovatelnosti • Použití technologií na serveru		Slabé stránky • Použité technologie na klientovi • Nutné instalovat JRE přesto že se jedná o webovou aplikaci • Podpora pouze PostgreSQL
Vnější prostředí	Příležitosti • Vznik nových dotačních programů • Lepší využití tržního potenciálu • Změna segmentace trhu • Využití lepších technologií		Hrozby • Nutná změna business plánu kvůli GPL licenci • Stávající konkurence na trhu • Zrušení stávajících dotačních programů • Využití zdrojových kódů konkurencí

Koha

Systém má více než 1100 instalací po celém světě. Je vyvíjený v jazyce Perl nad databází MySQL, lze jej tak provozovat pod operačními systémy Windwos, Unix. Jedná se o plně webový katalogizační systém, který běží na serveru Apache.

Tabulka 3: SWOT tabulka pro projekt Koha

Vnitřní prostředí	Silné stránky • Plně webová aplikace • Podpora OS Windows, Linux a Mac		Slabé stránky • Programováno v jazyce Perl • Použité technologie • Podpora pouze MySQL databáze • Zastaralý systém
Vnější prostředí	Příležitosti • Vznik nových dotačních programů • Lepší využití tržního potenciálu • Změna segmentace trhu • Využití lepších technologií		Hrozby • Nutná změna business plánu kvůli GPL licenci • Stávající konkurence na trhu • Zrušení stávajících dotačních programů • Využití zdrojových kódů konkurencí

Evergreen
Tento systém používají pouze stovky knihoven. Jedná se o desktopovou aplikaci programovanou v jazyce Perl a C, je provozovaný pouze nad databází PostgreSQL a operačními systémy Unix, Windows a Mac. Operační systém na serveru může být pouze Unix. Mezi jeho výhody patří možnost offline provozu.

Tabulka 4: SWOT tabulka pro projekt Evergreen

Vnitřní prostředí	Silné stránky • Podpora OS Windows, Linux a Mac • Podpora offline režimu • Systém vyvíjí skupina knihoven		Slabé stránky • Server lze provozovat pouze v linuxu • Desktopová aplikace • Programováno v jazyce Perl a C • Použití pouze PostgreSQL databáze
Vnější prostředí	Příležitosti • Vznik nových dotačních programů • Lepší využití tržního potenciálu • Změna segmentace trhu • Využití lepších technologií		Hrozby • Nutná změna business plánu kvůli GPL licenci • Stávající konkurence na trhu • Zrušení stávajících dotačních programů • Možnost zániku - málo používaný projekt • Využití zdrojových kódů konkurencí

Výběr vhodného open source projektu
Výběr vhodného open source projektu bude na základě vícekriteriálního rozhodování a SWOT analýzy každého projektu, následně bude probíhat volba mezi vybraným open source projektem a implementací celého systému od začátku.

Pro vícekriteriální rozhodování budou použita následující kritéria:

- uživatelská přívětivost – u tohoto kritéria je kladen největší důraz na prezentační vrstvu a framework, který byl zvolen. Toto kritérium je jedním z nejdůle-

žitějších, protože téměř všichni hodnotí systém podle jeho vzhledu a uživatelské přívětivosti. Je třeba poskytnou uživateli komfort na webu, který zná z desktopových aplikací,

- programovací jazyk, použité technologie – je dalším z nejdůležitějších kritérií, je preferován programovací jazyk Java a ostatní technologie použité v projektu,
- webový nebo desktopový systém – je důležitým kritériem, ale nejedná se o rozhodující kritérium. Výhodou je webová aplikace, ale zde hraje roli i to, jestli pro provozování webové aplikace je nutné provádět instalaci nějakého podpůrného programu,
- podporované OS – server – základním požadavkem je podpora operačních systémů založených na bázi Unix. Velkou výhodou je možnost provozovat server i pod operačními systémy Windows a Mac,
- podporované OS – klient – i když se může jednat o webovou aplikaci, ne vždy jsou všechny webové aplikace použitelné na všech operačních systémech. V některých případech je nutné provést instalaci různých podpůrných programů, bez kterých není možné systém provozovat. Za ideální by se dal považovat případ, kdyby se nemuselo nic instalovat a systém by mohl být provozován pouze v internetovém prohlížeči,
- počet instalací – toto kritérium je důležité z hlediska budoucnosti celého projektu, čím více má projekt instalací, tím je úspěšnější a lze předpokládat, že se bude dále rozvíjet,
- podporované databáze – čím více databází bude framework podporovat tím lépe, základním požadavkem je podpora MSSQL, Oracle a MySQL, všechny ostatní podporované databáze jsou výhodou, ale nejsou podmínkou. V rámci tohoto kritéria je brán v potaz i framework pro databázovou vrstvu,
- nadstandardní funkce – jedná se o funkce, které projekt poskytuje nad rámec požadovaných funkcí, může se jednat o podporu škálovatelnosti, offline režimu a podobně.

Z následující tabulky č. 5 zobrazující informace pro vícekriteriální rozhodování je naprosto zřejmé, že z daných variant je optimální použití frameworku NewGenLib.

Tabulka 5: Tabulka vícekriteriální rozhodování pro volbu open source frameworku jako základu celého systému

Kritérium	**Váha**	**Greenstone**	**NewGenLib**	**Koha**	**Evergreen**
Webový/desktopový	0,10	1	7	9	1
Podporované OS (server)	0,15	9	9	9	3
Podporované OS (klient)	0,05	6	9	9	9
Počet instalací	0,10	6	8	5	3
Uživatelská přívětivost	0,20	2	4	5	5
Programovací jazyk, použité technologie	0,20	4	7	4	5
Podporované databáze	0,10	2	7	2	2
Nadstandardní funkce	0,10	6	5	3	4
Suma	1,00	4,35	6,70	5,50	3,75

Jak ukazují SWOT analýzy předchozích open source projetků, tak analýzy vnější prostředí všech produktů jsou víceméně stejné. Kde se projekty výrazněji liší, tak je to v případě vnitřního prostředí, kde dochází k výraznějšímu odlišení jednotlivých projektů.

SWOT analýzy také ukazují spíše na projekt NewGenLib, kde jsou velmi dobré silné stránky tohoto projektu. Nejslabší stránkou tohoto projektu je technologie klientské části aplikace, tento problém mají však i všechny ostatní projekty. Řešení pomocí Java Appletů je v dnešní době již velmi zastaralé a pro jeho provozování je nutné instalovat na počítači JRE, proto v případě volby tohoto frameworku bude nutné nahradit celou prezentační vrstvu.

Na základě vícekriteriálního rozhodování a SWOT analýzy vychází za daných podmínek nejlépe projekt NewGenLib a proto při volbě mezi vývojem systému od začátku nebo použitím existujícího open source projektu bude právě NewGenLib zastupovat open source projekty.

6.1.3 Vlastní řešení

Implementace vlastním řešením má své výhody i nevýhody. Mezi výhody vlastního řešení lze uvést:

- velkou výhodou však je, že se aplikace tvoří od začátku a nemusí se brát ohled na již existující kód, který může být do značné míry omezující, nebo se musí upravit pro nové potřeby,
- další výhodou je, že si pro vývoj můžeme zvolit jazyk, který je pro naši firmu nejbližší. S tím souvisí znalost technologie, kterou lze ušetřit velké množství času, než kdyby se použil open source projekt v jazyce, který se ve firmě používá jen velmi málo,
- mezi další výhody vlastního řešení je to, že tento systém je již od začátku navrhovaný jako obecný katalogizační systém, naproti tomu jsou již existující open source projekty, které jsou striktně zaměřeny na knihovny. Výsledkem nemá být knihovní systém, ale obecný katalogizační systém, který se řídí knihovními standardy, tyto standardy jsou pevně dané a dodržované po celém světě,
- při návrhu vlastního řešení navíc můžeme plně uplatnit znalosti a zkušenosti kooperující firmy LANius, tak i své znalosti a zkušenosti.

Mezi nevýhody lze zařadit:

- největší nevýhodou je časová náročnost (a tím i finanční) vlastní implementace – obecně tato možnost bývá nejdražším způsobem vývoje nového produktu. Jedná se o opravdu velkou nevýhodu a je třeba pořádně zvážit, jestli se tato možnost do budoucna vyplatí,
- v některých případech „se vymýšlí kolo“ – mohou existovat projekty, které vyhovují všem požadavkům, mohou být na vysoké úrovni a mohou být poskytovány zdarma, ale přesto jsou projekty navrhovány a implementovány znovu od začátku.

V rámci tohoto řešení je nutné udělat vše od úplného začátku. Musí se vytvořit UML diagramy, které budou popisovat systém a umožní konzultace, jestli je systém navrhovaný správně.

Součástí implementace vlastním řešením je volba programovacího jazyka i ostatních frameworků, volba podporovaných databází, operačních systémů, internetových prohlížečů apod. Při implementaci je nutné navrhnout testování aplikační i prezentační vrstvy.

6.1.4 Odhad a srovnání nákladů obou řešení

Všechny odhady budou uvedeny v univerzální jednotce pro plánování - man-day. Tento termín označuje množství práce vykonané jedním pracovníkem za jeden den.

Pro systém Clavius již byl vytvořen webový katalog a Z39.50 server, proto mohou být časové odhady pro vlastní implementaci považovány za reálné. Odhad délky vlastní implementace byl proveden po kompletním prostudováním katalogizačního systému Clavius. Odhady byly konzultovány a upravovány s autory systému Clavius (nově navrhovaný systém má být přímou náhradou tohoto systému), tak aby byly co nejvíce reálné.

Časové odhady pro implementaci open source projektu byly provedeny po prostudování tohoto systému. Největší prostor byl věnován studiu návrhu open source systému, aby byl zjištěn potenciál pro jeho rozvoj. Byly také prozkoumány zdrojové kódy jak z hlediska čitelnosti, pojmenování metod a proměnných, tak také z dokumentační stránky. Jelikož je open source framework implementován v jazyce Java a jsou v něm použité velmi dobře známé technologie jako je Hibernate, tak je možné vytvořit časové odhady reálně.

V tabulce jsou zobrazeny implementační náročnosti jednotlivých variant. Všechny časové náročnosti budou zobrazeny v univerzální jednotce man-day (MD).

Tabulka 6: Implementační náročnosti jednotlivých variant

Funkce	**NewGenLib**	**Vlastní implementace**
Základ aplikace	20	150
Katalogizace	100	300
Akvizice	20	200
Webový katalog	100	100
Výpůjční protokol	20	250
Revize	10	50
Z39.50 server	5	35
Z39.50 klient	5	10
Použití GWT pro klientskou část	100	0
Lokalizace	20	0
Suma	400	1 200

6.1.5 Volba vhodného řešení pro základ aplikace

SWOT tabulky č. 2 a č. 7 popisují vnitřní a vnější prostředí projektu NewGenLib a vlastní implementace celého systému od začátku. SWOT tabulka byla rozšířena o položky, které jsou velmi důležité pro porovnání s vlastním řešením.

Tabulka 7: SWOT tabulka vlastního řešení

Vnitřní prostředí	Silné stránky • Software přesně podle požadavků • Obecný katalogizační systém se zaměřením na knihovny • Využití know-how firmy LANius • Žádné omezení licencí		Slabé stránky • Orientace pouze na středoevropský trh • Nový produkt – žádné reference • Nákladná implementace
Vnější prostředí	Příležitosti • Využití open source frameworků • Možnost prodeje licence, nejen forma služby • Vznik nových dotačních programů • Možnost outsourcování vývoje		Hrozby • Stávající konkurence na trhu • Pozdní příchod na trh – vznik nové konkurence • Omezení státních dotací (pro ČR – VISK apod.) • Možnost špatného návrhu systému

Požadavky na systém

Jedním z požadavků na nový systém je, že se má jednat o obecný katalogizační systém. Zvolený open source projekt je navržený a implementovaný jako čistě knihovnický systém, který je vyvíjený zaměstnanci knihoven. Při implementaci vlastním řešením lze systém navrhnout a implementovat jako obecný katalogizační systém, do kterého lze ukládat libovolné informace a následně v nich vyhledávat.

Důležitým požadavkem je ukládání velkého množství záznamů. Open source systém používají spíše menší knihovny, žádná velká knihovna tento systém nepoužívá. Systém nepodporuje studii FRBR, která má umožňovat lepší vyhledávání ve velkém množství záznamů.

Dalším požadavkem je podpora multi-knihovnosti. Zvolený open source projekt tento požadavek nesplňuje, muselo by dojít k výrazné úpravě tohoto projektu. V případě vlastního řešení lze s tímto požadavkem počítat již na začátku a průběžně jej implementovat do systému.

Licence

Jelikož byla prakticky jedinou možnou volbou open source projektu s licencí GPL, pak by nově vzniklý systému musel být vydán také pod licencí GPL. U projektů vydaných pod licencí GPL je ještě možnost odkoupit celý systém pod jinou licencí, protože autoři původního systému mohou vydat software pod libovolnými licencemi. Nákup systému by byl velmi obtížný a také velmi nákladný. Tato varianta by ztrácela výhodu open source řešení, spíše by se jednalo o koupi systému se zdrojovými kódy.

Pokud by se použila licence GPL, pak by si tento projekt mohl kdokoli stáhnout a bezplatně upravovat a také poskytovat systém jako službu veřejnosti. Současně s tím, by nově vzniklý systém nemohl být prodáván. S volbou open source pod licencí GPL by se u produktu přišlo o možnost prodat zákazníkovi výsledný systém. Musela by se zvolit jiná obchodní strategie – např. poplatky za instalaci, hostování a servis. Některé firmy nemohou mít z bezpečnostních důvodů systém hostovaný, protože systém může být napojen do stávající infrastruktury podniku na ostatní systémy, které nemají přístup k internetu, nebo jejich vnitřní předpisy neumožňují instalaci mimo sídlo společnosti. Proto u těchto zákazníků by nebylo možné instalaci provádět, naopak pokud by byla provedena vlastní implementace software byly by tyto instalace velmi výhodné.

U vlastní implementace lze využít všech výhod cloudového řešení a v případě nutnosti prodat katalogizační systém jako produkt.

Náklady implementace

Obecně platí, že vývoj vlastního systému je nejdražším způsobem implementace. Počáteční časové odhady na implementaci vlastního a úpravou open source projektu ukazují na volbu open source projektu, protože časové nároky jsou o více než 65 % nižší než na implementaci vlastního řešení.

Náklady při vývoji vlastního systému by se měly postupem času snižovat, protože celý systém je navrhovaný a implementovaný podle současných požadavků.

Technologie

U open source projektu jsou některé technologie zvolené vhodně a některé jsou velmi nevhodné.

Použité technologie pro klientskou část aplikace je v dnešní době již zastaralé a nevhodné. Technologie použité pro webový katalog jsou také nevhodné a také návrh katalogu neodpovídá požadavkům.

Pro přístup k datům je použitá technologie Hibernate, což je v dnešní době nejpopulárnější framework v Java aplikacích pro přístup k databázi. Klientská technologie GWT je sice na databázovém frameworku nezávislá, ale pro tuto technologie je pro práci s databází vhodnější technologie Ujorm.

Další technologie jako Solr pro vyhledávání, JasperReports pro vytváření reportů byly zvoleny vhodně.

Velkou výhodou vlastní implementace je to, že se pro každou část aplikace, každý požadavek se může zvolit ta nejvhodnější technologie.

Rozhodnutí

Výsledný systém nebude projektem pro konkrétního zákazníka, ale produktem, který se bude nabízet od těch nejmenších knihoven až po ty největší. Nově vznikající systém, jak již bylo několikrát uvedeno, má být přímou konkurencí systému Aleph. Tento systém používají velké knihovny a na tento systém jsou kladeny velmi vysoké nároky. Všechny open source projekty byly navrhovány pro malé a střední knihovny (jendá se o knihovní systémy), nebyla nalezena žádná velká knihovna, která by open source projekt používala. V současnosti by byl vývoj určitě rychlejší s volbou open source projektu, ale do budoucna by bylo velmi obtížné, časově a finančně náročné upravit aplikaci pro práci s velkým množstvím záznamů, případně na speciální požadavky velkých knihoven.

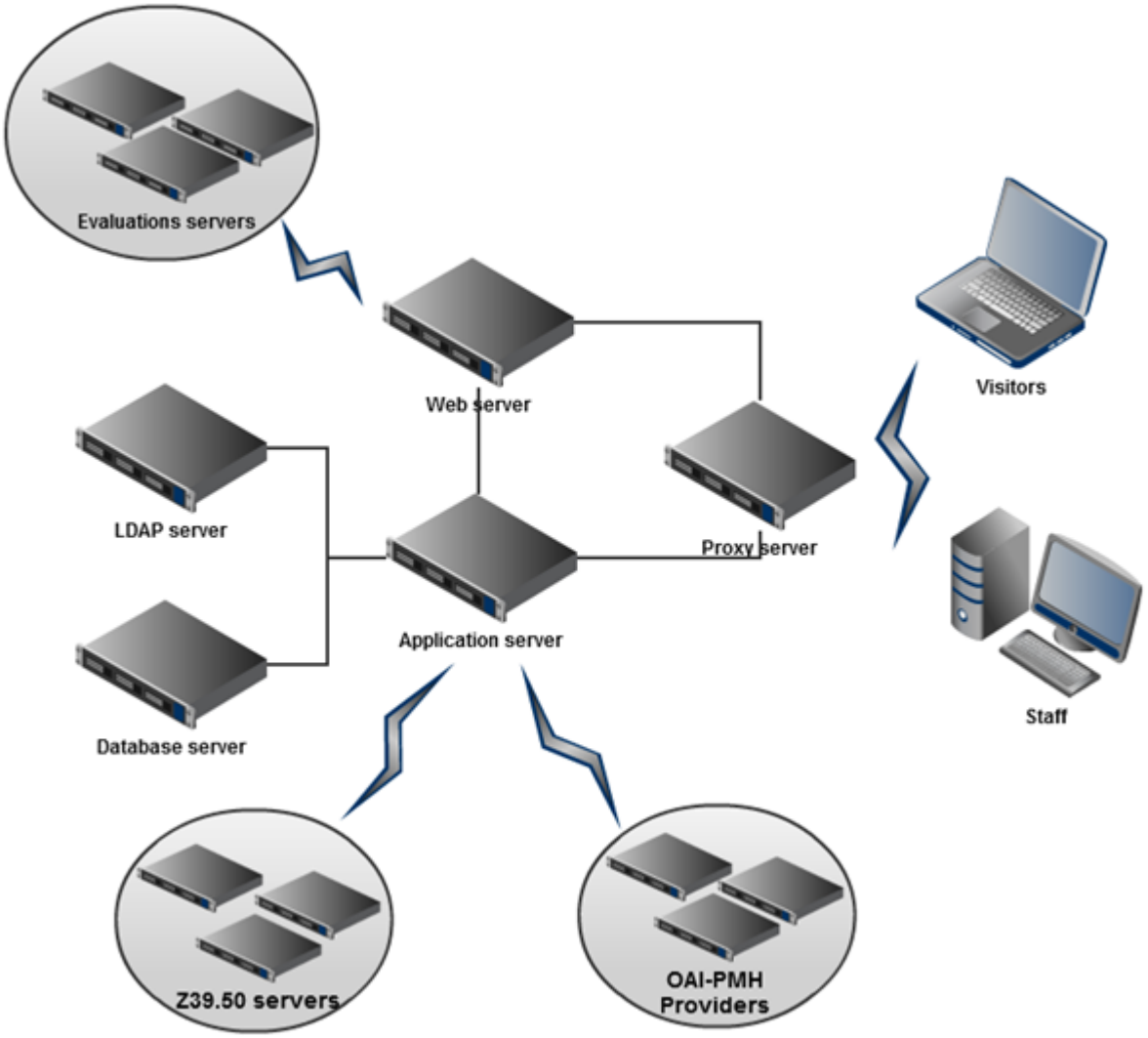
Dva nejdůležitější faktory jsou cena a licence. Jak již bylo uvedeno v tabulce č. 6, tak cena ukazuje na volbu open source projektu, kdy lze ušetřit více než 65 % nákladů. Při této volbě by však musel být výsledný produkt vydaný také pod licencí GPL, což je druhý ze dvou nejdůležitějších faktorů. Tento způsob licence je velmi omezující. Mezi další velké nevýhody lze zařadit, že open source systém je čistě knihovní systém. Další nevýhodou je, že z celého systému by zůstal pouze základ aplikace, protože modul webového katalogu i klientská část aplikace by musely být kompletně vyměněny, dále by se musely upravit ostatní funkce dle požadavků.

Bylo přihlédnuto ke všem silným a slabým stránkám, příležitostem a hrozbám (na základě tabulky č. 2 a tabulky č. 7). Byly také zváženy předchozí faktory i vlastní zkušenosti a pocity. Na základě všech těchto faktorů bylo rozhodnuto v implementaci systému vlastním řešením a využití znalostí a zkušeností kooperující firmy LANius, které na trhu knihovních systémů působí dvacet let. I když budou investice zpočátku výrazně vyšší než v případě open source projektu, s postupem času by mělo dojít ke změně a tyto vynaložené investice by se měly rychle vrátit, protože bude možné oslovit zákazníky, které by s open source systémem vydaným pod licencí GPL nebylo možné oslovit.

6.2 Technologický návrh aplikace

6.2.1 Architektura

Systém bude napojen na Z39.50 servery a OAI-PMH providery, které může prohledávat a přebírat od nich data. Jelikož systém může používat LDAP autentizaci, tak je tento způsob do schématu také zahrnut.
Následující obrázek č. 6 popisuje rozložení jednotlivých částí systému. Nejdůležitější myšlenkou je oddělení vlastního katalogizačního systému od prezentační části sytému (tzv. OPAC – Online Public Access Catalog). Vlastní katalogizační systém slouží pro zaměstnance a katalog je zaměřený zejména na čtenáře.



Obrázek 6: Architektura systému

Webový katalog by měl být oddělený od vlastního katalogizačního systému a z těchto následujících důvodů:

- využití co nejlepších technologií pro každou oblast. Webový katalogizační systém se bude co nejvíce přibližovat desktopové aplikaci a pro tento případ by měla být zvolena vhodná technologie. Webový katalog pro procházení záznamů čtenáři musí být co nejrychlejší, přehledný a jednoduše skinovatelný, proto i v tomto případě by měla být zvolena ta nejvhodnější technologie,
- téměř 100% dostupnost webového katalogu. Někdy může být problematické nasazování nových verzí základu systému. Při nasazování nové verze mohou přesto čtenáři používat webový katalog bez omezení, i když vlastní katalogizační systém slouží jako přístupové rozhraní k datům,
- existence jednoho rozhraní pro externí systémy. Bude existovat pouze jedno rozhraní pro „veřejné katalogy“. Veřejnými katalogy je myšleno zejména klasický webový katalog ke kterému uživatel přistupuje zejména s použitím počítače a internetové prohlížeče. Dalšími možnostmi jsou katalogy optimalizované pro mobilní zařízení – tzn. jak mobilní internetové prohlížeče, tak přímo aplikace pro mobilní telefony,
- možnost zásadních změn ve webovém katalogu, kdy změny nemusí probíhat pouze na úrovni kaskádových stylů, ale mohou probíhat přímo ve zdrojových kódech webového katalogu,
- možnost provozování více webových katalogů nad jedním katalogizačním systémem, vlastní katalogizační systém pak působí jako přístupové rozhraní k datům. Tuto vlastnost lze použít v případě, kdy knihovna potřebuje mít odděleny webové katalogy a zobrazovat různá data pro čtenáře z interní sítě a uživatele z internetu,
- bezpečnostní důvody. Některé firmy budou požadovat možnost provozovat katalogizační systém na lokální síti a do internetu mít vystaven pouze webový katalog. Již při návrhu a implementaci je jasně zřejmé, které funkce bude webový katalog podporovat a jaké funkce mají být vystaveny prostřednictvím webových služeb. Lze pak jednoduše nakonfigurovat proxy server, který umožní přístup například pouze do webového katalogu.

Výhodou navrhovaného řešení je, že v případě instalace pro libovolnou střední knihovnu lze nainstalovat vše na jeden server, nebo vše dodat již v nakonfigurovaném virtuálním serveru, což zjednoduší celou instalaci a počáteční nastavení.

V případě velké knihovny lze využít stávající infrastruktury, jako jejich databázového, LDAP, aplikačního a nebo proxy serveru. V takovém případě lze přímo oddělit katalogizační systém od webového katalogu a přístup do katalogizačního systému povolit pouze přes VPN. Malé knihovny budou provozovány formou hostované aplikace a proto rozložení výkonu (rozdělení fyzických serverů) bude podobné jako v případě jedné velké knihovny mající samostatnou instanci systému.

6.2.2 Technologie

Jako programovací jazyk základu celého katalogizačního systému byl zvolen robustní programovací jazyk Java. Podle Pragmatic Progamers (2006) je vývoj v programovacím jazyce Java nejdelší, ať už je srovnání provedeno mezi statickými, nebo dynamickými jazyky. Tato technologie má však velké výhody oproti ostatním programovacím jazykům.

Podle Tiobe indexu je Java nejpoužívanější programovací jazyk, v tomto jazyce také existuje pravděpodobně největší množství open source frameworků, které pak lze použít pro vývoj aplikací (TIOBE Software, 2012). Dále nabízí velmi dobré možnosti debuggování, napojení na ostatní programovací jazyky apod. Omezení tohoto jazyka jsou široce známy, existuje nespočetné množství různých fór a diskuzí, kde jsou problémy řešeny. Proto je tento jazyk preferován.

Celý systém bude navrhován jako webová aplikace. Jako klientská technologie pro katalogizační systém je zvolena technologie GWT, protože se tato technologie nejvíce blíží možnostem desktopové aplikace a je programována právě v jazyce Java. Tuto technologii lze vylepšit o komponenty Ext GWT, které vytváří uživatelsky velmi přívětivé systémy, které jsou postaveny na robustním základu. Ext GWT zároveň zaručuje funkčnost ve všech hlavních internetových prohlížečích. Je však nutné koupit licenci k použití těchto komponent, základní cena je necelých $600. S použitím těchto komponent se ještě více minimalizuje časová náročnost na programátorské práce pro ladění systému pro jednotlivé internetové prohlížeče.

Jako rozhraní webového katalogu je zvolena technologie PHP, protože se v dnešní době jedná o nejpoužívanější technologii pro tvorbu webových stránek, existuje pro ni velké množství technologií a hlavně vývoj v této technologii je velmi rychlý. Tím, že bude webový katalog oddělen, je jednoduše možné technologii PHP v budoucnu vyměnit za Python, Ruby apod. Mezi velkou výhodu lze jistě zařadit možnost nasazení nové verze bez nutnosti restartování webového serveru, to výrazná výhoda oproti technologiím jako jsou JSF (webové aplikace psané v jazyce Java) apod., kdy při nasazení nové verze musí dojít k zastavení celého aplikačního serveru a nahrazení WAR souboru.

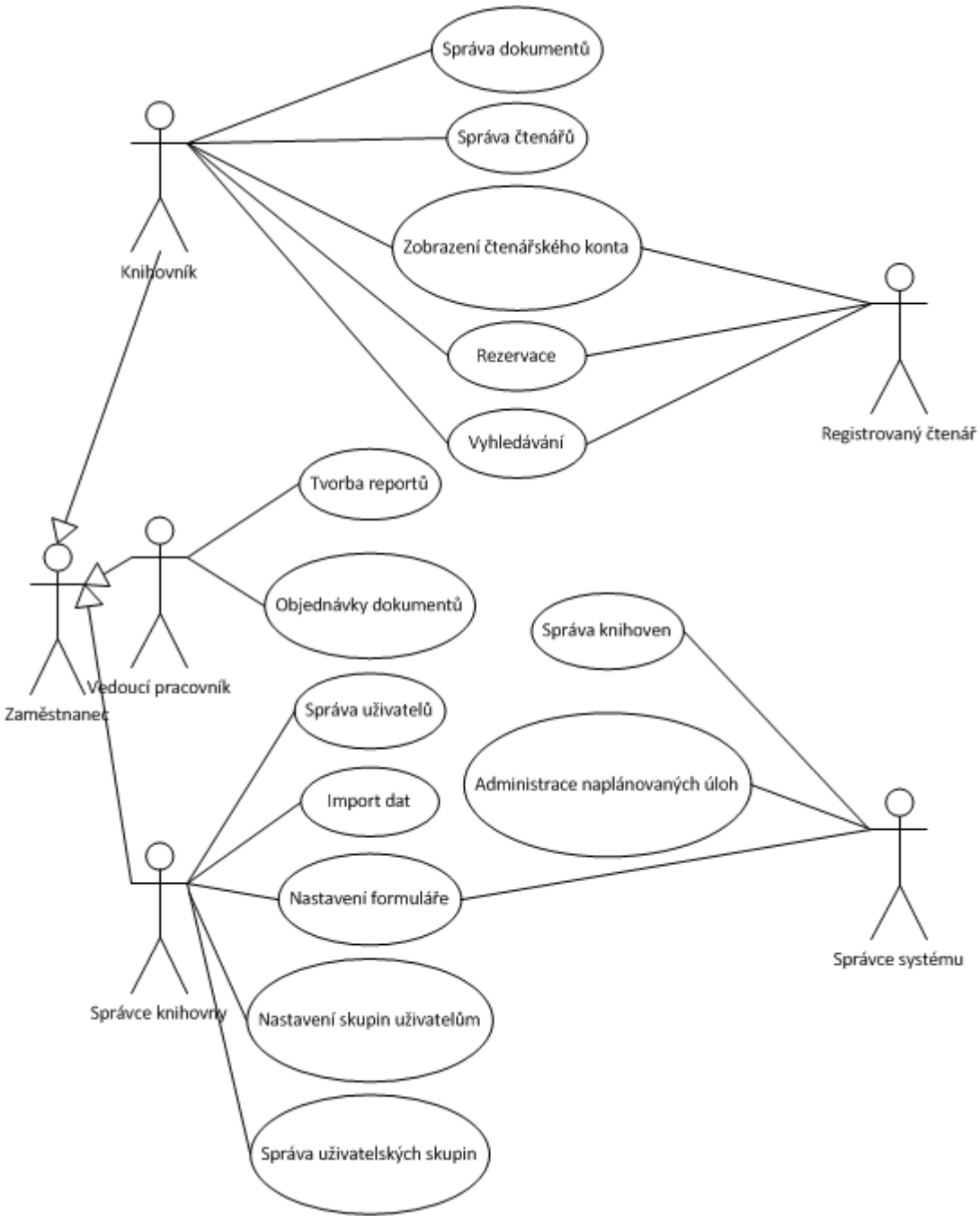
Systém bude podporovat tři základní typy databází. Jedná se o Oracle, MSSQL a MySQL. V případě, že použijeme jakýkoli framework, který podporuje rámcově

desítky databází, pak je nutné stejně dbát na to, že tento framework nevyřeší vše sám. Je to hlavně proto, že jsem z vlastních zkušeností doposud neviděl framework, který by bezchybně podporoval zmíněné databáze. Dalším důvodem je fakt, že se bude zpracovávat velké množství dat a pro každou databázi musí být dotazy napsané trochu jinak, aby byly výkonově co nejlepší. Takový framework je možné použít pouze pro administrační část, typicky CRUD tabulky.

6.2.3 Analytický model aplikace

Následující Use case diagram zobrazuje elementární části katalogizace, výpůjčního protokolu, akvizice a webového katalogu. Dále je zobrazeno administrátorské nastavení knihovny i celého systému. V tomto diagramu jsou rozlišeni aktoři podle toho, jakou mají roli v knihovně, jedná se o knihovníky, administrativní pracovníky a správce systému (v rámci knihovny). Dalšími aktory jsou čtenáři (ať už přihlášení nebo nepřihlášení) a také správci celého katalogizačního systému, kteří jsou nezávislí na knihovně.

Na use case v příloze obrázku č. 16 jsou vidět následující aktoři: Zaměstnanci knihovny, Čtenáři a Správci systému. U čtenáře se rozlišuje jestli je přihlášený nebo ne, to má vliv na dostupné funkce pro daného čtenáře. Zaměstnanci knihovny se rozdělují na tři základní typy, a to jsou knihovníci, vedoucí pracovníci a správci knihovny. Jedná se o obecné rozdělení pozic, v každé knihovně jsou role upraveny trochu jinak, proto musí existovat dynamická tvorba rolí uživatelů v systému. Každý informační systém potřebuje svého správce, který je nezávislý na knihovně a spravuje systém jako celek (provádí různé optimalizace, správu společného nastavení, předčasné spouštění naplánovaných úloh a pod.), v tomto use case diagramu je to aktor správce systému.

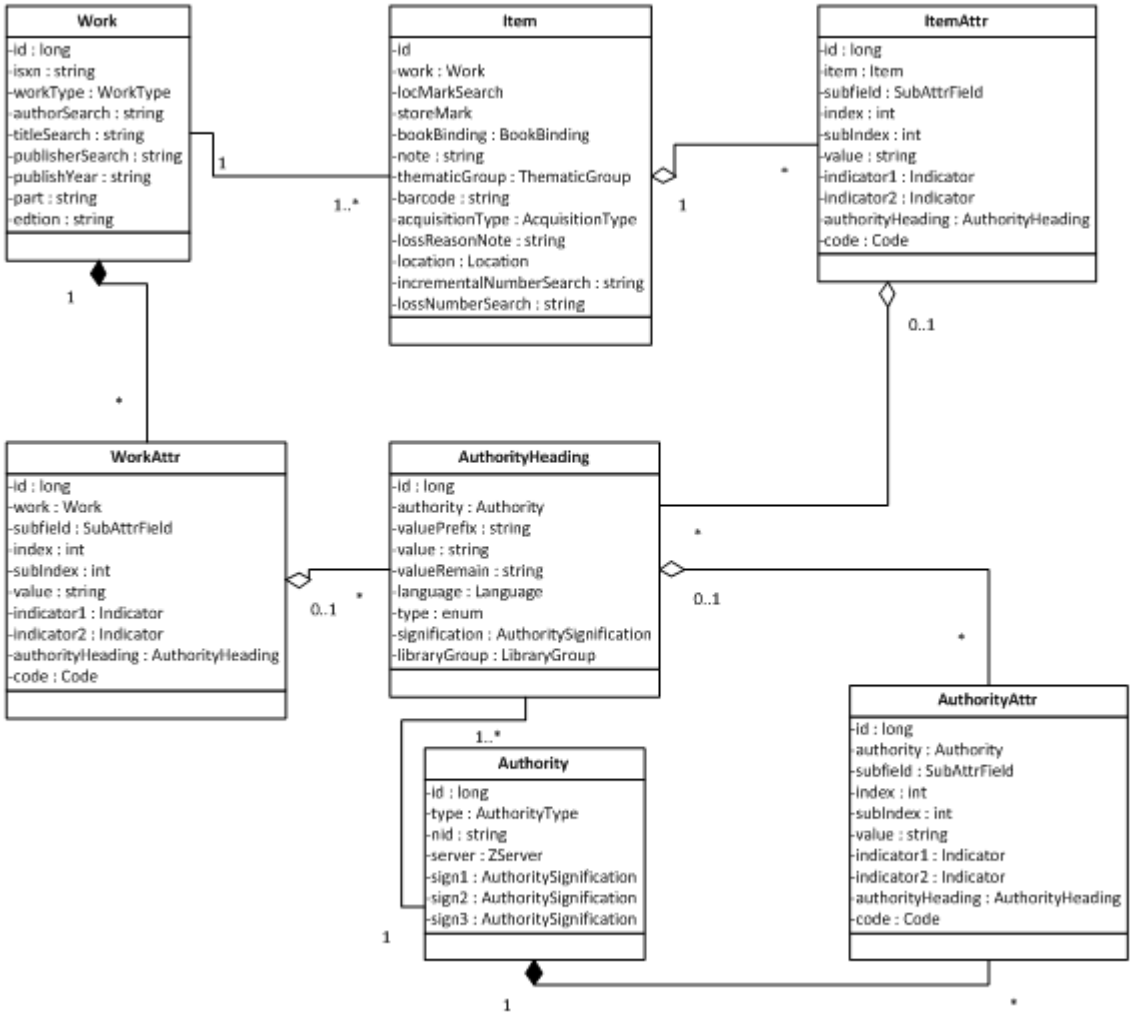


Obrázek 7: Use case diagram

V následujícím diagramu tříd jsou (tabulka č. 8) zobrazeny požadavky na ukládání více než milionu informací k jednomu dílu (autoritě, svazku) s libovolnou délkou, ukládání nadřízených, podřízených titulů a vazeb mezi tituly na stejné úrovni, dále jsou v návrhu zobrazeny požadavky podle studie FRBR.

V případě ukládání autorit jsou zohledněny požadavky na vazby mezi autoritami, které umožňují ukládat nadřízené, podřízené vazby mezi autoritami. Bylo také

myšleno na ukládání vyloučených tvarů, jazykových mutací a odkazů mezi autoritními záhlavími.



Obrázek 8: Diagram tříd

Podrobnější diagram tříd je uveden v příloze viz obrázek č. 17.

6.3 Strategie pro vybranou dílčí část

Jako dílčí část systému byla vybrána jedna z nejdůležitějších oblastí. Touto oblastí je indexování a vyhledávání evidovaných dat. V případě vyhledávání se největší důraz klade na rychlost a přesnost vyhledávání.

Celý systém je navrhovaný tak, aby zvládal i velké množství dat. Předpokládá se s prohledáváním dat, které budou v jednotkách GB. Nejčastější však bude případ prohledávání v jednotkách desítek GB. Jelikož je systém navrhovaný tak, aby

umožňoval provoz v cloudových centrech, pak budou nastávat případ, kdy se budou prohledávat data v řádech stovek GB. Proto indexační a vyhledávací nástroj musí podporovat škálovatelnost.

Při vyhledávání je nutné brát v úvahu fakt, že aplikace bude prohledávat téměř všechna data z databáze. Framework musí dále splňovat následující požadavky:

- rychlost – u frameworku můžeme rozlišovat dvě rychlosti – rychlost vyhledávání a indexování:
  - rychlost vyhledávání - je jeden z nejdůležitějších faktorů, a proto při volbě bude plnit klíčovou funkci. Nejedná se o rychlost vyhledání a zobrazení záznamů, o zobrazení se mohou starat různé technologie a proto rychlost od vyhledání až po vykreslení uživateli je značně zavádějící. Nejdůležitější je čas od předání dotazu vyhledávácímu frameworku až po získání výsledků z tohoto frameworku,
  - rychlost indexování – rychlost indexování má spíše vedlejší význam, nesmí se však na tento faktor zapomenout. Data se budou indexovat v průběhu ukládání záznamů a v tomto případě je celkem jedno, jestli se změna projeví za 0,1 vteřiny nebo za vteřinu. Rychlost indexace je nutné brát v úvahu při importu rozsáhlých dat, nebo při reindexaci celé databáze.
- řazení výsledků - u řazení je opět nutné rozlišovat na řazení podle relevance a podle abecedy. Toto jsou další dvě klíčové funkce:
  - řazení podle relevance – umožňuje seřazení výsledků podle důležitosti tak, aby se uživateli zobrazil hledaný výsledek jako jeden z prvních. Framework musí podporovat nastavení vah pro určitá vyhledávaná pole. Například název nebo autor musí mít vyšší váhu, než například místo vydání. Vyhledávaným polím musí podporovat změnu váhy a musí být dynamicky přidávána, upravována za běhu systému,
  - řazení podle abecedy – na první pohled tento požadavek vypadá jako samozřejmost, ale není tomu tak. V knihovních systémech je velmi důležité správné řazení, proto framework musí podporovat UTF8. Jako problematické znaky lze označit české (ale i jiné neanglické znaky), jako jsou například písmena „č“, „ž“, „ň“, ale také „ch“.
- vyhledávání se speciálními znaky (wildcards) – také bývá označováno jako divoké znaky, lze mezi ně zařadit hvězdičku, uvozovky, otazník, logické operátory apod. Jsou to znaky, které umožní uživateli zmenšit počet vyhledaných výsledků, nebo počet naopak zvětšit. Každopádně tyto znaky mají uživateli

umožnit nalezení požadovaných informací. Například pokud uživatel neví, jak se přesně píše autorka „Marianne Curley“ (jestli se používá jedno písmeno „n“ nebo více), pak lze použít operátor hvězdička a hledat např. „Marian* Curley“. Nebo uživatel zná pouze nějakou frázi nebo chce zobrazit všechny díly jedné knihy, pak je možné použít uvozovky pro vyhledání určité fráze např. „Strážci času“.

- vyhledávání – framework musí podporovat vyhledávání ve veškerých indexovaných záznamech (ve stylu „Google“), ale také vyhledávání v určitých polí.
  - Vyhledávání ve všech polích většinou vrací tisíce záznamů a pokud je dotaz položen špatně (uživatel neví co přesně hledá, nebo zná jen málo informací), pak se mu data nemusí zobrazit na prvních pozicích. Při vyhledávání ve všech polí je nutná podpora vyhledávání podle relevance a správné nastavení vah ve vyhledávaných polích.
  - Je nutné podporovat vyhledávání pouze v určitých polích – například uživatel ví, že v názvu knihy bylo slovo „auta“, proto by měl mít možnost vyhledávat pokročilým způsobem, ve kterém si určí, ve kterém poli má být hledaný výraz vyhledán.
  - AND a OR operátory – framework musí podporovat oba operátory jak pro vyhledávání mezi poli, tak také v rámci jednoho vyhledávacího pole jako operátor mezi jednotlivými slovy (tokeny). Lze to uvést na příkladu kdy se vyhledává ve všech polích dotaz „Smrt krásných srnců“, tak mezi jednotlivými tokeny musí jít nastavit operátor AND nebo OR. Další možností je pokročilejší způsob hledání, kdy se hledá v poli s rokem vydání rok „1987“ a v názvu „Babička“. Pak je rozdíl jestli se vyhledává kniha s rokem „1987“ nebo názvem „Babička“, nebo kniha s rokem „1987“ a název „Babička“.
  - Case sensitive – v některých případech je vhodné rozlišit velikost písmen, jestli se vyhledávaná slova přesně shodují se indexovanými výrazy. Toto může být vhodné například při frázovém vyhledávání nebo pro zvyšování relevance výsledků.
  - Diakritické vyhledávání – framework by měl umožňovat vyhledávání s rozlišením diakritiky, aby se nemuselo vytvářet nové vyhledávané pole, které bude uchovávat data bez diakritických znamének. Toto vyhledávání je vhodné pro uživatele používající nestandardní klávesnici pro danou zemi, nebo pro vyhledávání z tabletů a mobilních telefonů, kde psaní diakritických znamének může trvat déle.

  - Nastavení vah vyhledávaným polím – každé vyhledávané pole musí podporovat nastavení váhy pro vyhledávání. Čím přesněji půjde váha nastavit, tím lépe. Například nastavení váhy stylem vysoká, střední, nízká je zcela nevyhovující. Při vyhledávání je rozdíl, pokud se vyhledávané slovo najde v názvu nebo místu vydání titulu.

- omezování vyhledaných výsledků - bývá také označováno jako „facetování“. Tento způsob umožňuje omezit vyhledané výsledky, podle daných kritériích. U každého kritéria by bylo vhodné zobrazit počet výsledků, které budou zobrazeny po volbě tohoto kritéria. Mělo by být umožněno zvolit si více kritérií pro jedno vyhledávané pole i napříč více vyhledávanými poli. Může být hledáno například „informační systémy“ a kritérii omezit pouze na dokumenty typu diplomové práce, které byly napsány roce 2010 nebo 2011. Tím se výrazně omezí počet výsledků a je možné tak rychleji dohledat požadované záznamy. Tato možnost bude nejpoužívanější u vyhledávání přes všechna pole, kdy je výsledkem vyhledávání nejvíce výsledků, může být však používána i pro ostatní typy hledání,

- ohýbání slov – je označováno také jako „lemmatizování výrazů“. Tato funkce není podmínkou frameworku, vyhledávání lemmatizovných výrazů lze zařídit i na straně aplikace, kdy budou indexovány pouze kořeny slov. Tato funkce uživateli může výrazně zpříjemnit vyhledávání a umožnit nalezení výsledků, které by bez této funkce v žádném případě nenašel. Jako příklad lze uvést, kdy uživatel přesně neví, jak se kniha jmenuje, pouze tuší, že název obsahuje něco o „krásném srnci“ a ještě se tam vyskytuje slovo „smrti“. Může být pak vyhledán výraz „krásném srnci smrti“ nebo „o smrti krásného srnce“, ale uživatel měl s největší pravděpodobnosti na mysli knihu „Smrt krásných srnců“. Pokud by systém neumožňoval ohýbání slov, uživatel by pravděpodobně tuto knihu sám nikdy nenašel,

- zvýraznění vyhledaných a lemmatizovaných výrazů – většinou je uživatel častým návštěvníkem webu knihovny a ví, že na prvním místě se zobrazuje název díla. Po vyhledání a zvýraznění se pak může rychleji rozhodnout, jestli pro něj výsledek má, nebo nemá význam. Uživatel může hledat výraz „Němcová“ (zajímá ho například kniha s názvem „Naše paní Božena Němcová“), pak může podle zvýraznění na první pohled rozlišit, jestli našel to co hledá, aniž by musel číst všechny názvy – pokud název nebude zvýrazněn vůbec, pak uživatele tento záznam vůbec nezajímá a může pokračovat dále. Toto zvýraznění výsledků pak může výrazně zkrátit čas od položení dotazu až po nalezení požadovaného výsledku,

- seskupení údajů (groupovací funkce) – je to stejné funkce, která se používá v SQL dotazech, kdy je nutné seskupit výsledky vyhledávání. Existují případy, kdy jeden titul má více svazků (vazba 1:N) a uživatel vyhledává informace ze svazků a výsledkům může vyhovovat více svazků, bylo by pak špatně zobrazit uživateli daný titul vícekrát. Proto je nutná podpora této groupovací funkce,
- škálovatelnost – v případě, že se v systému bude uchovávat opravdu velké množství dat, musí se nad těmito daty umět rychle vyhledávat. V takových případech již nemusí stačit výkon jednoho serveru a proto by měl framework podporovat rozložení indexu na více serverů, čímž dojde k zvýšení výkonu a tím i rychlosti vyhledávání.

6.3.1 Open source framework

Vyhledávání bylo zaměřeno na frameworky pod licencí LGPL a Apache licence, Version 2.0. Právě pod touto verzí je technologie Lucene, se kterou mám výborné zkušenosti. Tento framework byl použit v jednom projektu pro indexování a vyhledávání v dokumentech typu .doc (Microsoft Word). Technologie dokázala prohledávat tisíce rozsáhlých souborů ve velmi krátkém čase. Ve skutečnosti technologie neprohledává přímo dokumenty, databázi apod, ale při vkládání nového dokumentu si obsah ukládala do vlastního indexu, ve kterém pak vlastní vyhledávání probíhá. Vyhledávání je pak velmi rychlé a je možné takto indexovat a prohledávat prakticky jakákoli data. V tomto případě se velmi osvědčila jak z hlediska rychlosti implementace, tak i rychlosti indexace a vyhledávání.

Pro potřeby využití webového katalogu však není dostatečná, této technologii například chybí možnost facetování. Existuje však technologie Solr, což je nadstavba nad Lucene. Tato technologie je vydána také pod verzí Apache License, Version 2.0 a tak tato licence nemá žádný vliv na výslednou licenci vyvíjeného systému. Technologie je na nastavení o něco složitější, než technologie Lucene, ale díky tomu poskytuje více funkcí.

Solr má za sebou dlouhou historii, existuje již 11 let a za tu dobu se neustále vyvíjí a zrychluje. Výhodou této technologie je, že pro ni existuje program (Luke-all), který umožňuje prohledávat přímo jeho index, zobrazovat obsah – informace k jednotlivým indexovaným záznamům. K výhodám této technologie také patří možnosti nastavení indexace, v některých případech je vhodné do indexu ukládat přímo i data, ty se mohou v některých případech hodit – např. jako našeptávač.

Framework Solr splňuje téměř všechny požadavky na vyhledávací a indexační server až na jeden drobný nedostatek. Tento framework má problém se zvýrazněním lemmatizovaných výrazů, proto v případě této volby bude ještě nutné přidat čas potřebný k implementaci vlastního highliteru.

Jelikož tento framework se jeví jako vyhovující, bylo na něm provedeno testování rychlosti vyhledávání. Testování bylo prováděno na běžném vývojovém počítači o parametrech:

- 8 GB DDR3 RAM, 1 333MHz
- 500 GB HDD 7200 RPM, 4.2ms (Windows NTFS)
- Intel Core i5-2500 Quad-Core

Bylo indexováno 200 000 titulů, 250 000 svazků a 3 500 000 titulových atributů. U jednoho titulu bylo v průměrů 17,5 titulových atributů. Titulovým atributem je například myšleno – název, autor, rok vydání, místo vydání, nakladatel, hlavní autor, vedlejší autor, ISBN, rozsah apod. V následující tabulce č. 8 jsou zobrazeny jednotlivé dotazy a u nich je uveden čas v milisekundách.

Tabulka 8: Tabulka zobrazující časy hledání frameworku Solr

Vyhledávaná fráze	**Čas vyhledávání (ms)**
Jedno slovo s hvězdičkou (začíná)	90
Jedno slovo	25
Tři slova	40
Tři slova a jedno z nich s hvězdičkou (začíná)	170
Lorem Ipsum is simply dummy text of the printing and typesetting industry	70
Dvouslovné exaktní vyhledávání (s uvozovkami)	15

I přesto, že vyhledávání bylo testováno na vývojovém počítači, tak jsou rychlosti vyhledávání velmi dobré. Lze předpokládat, že na serverech s Unixovým operačním systémem budou rychlosti vyhledávání podstatně vyšší, a to z několika důvodů – serverové komponenty jsou výrazně výkonnější než desktopové a také z důvodu rychlosti oddílu zformátovaných na NTFS (na kterém bylo prováděno testování).

Framework mimo jiné nabízí další užitečné funkce:

- rozšíření Lucene Query Language,
- geoprostorové vyhledávání,
- pokročilé a konfigurovatelné textové analýzy,
- vysoce konfigurovatelné a uživatelsky rozšířitelné cachování,
- další výkonnostní optimalizace,
- možnost indexace a vyhledávání v PDF, Word, HTML.

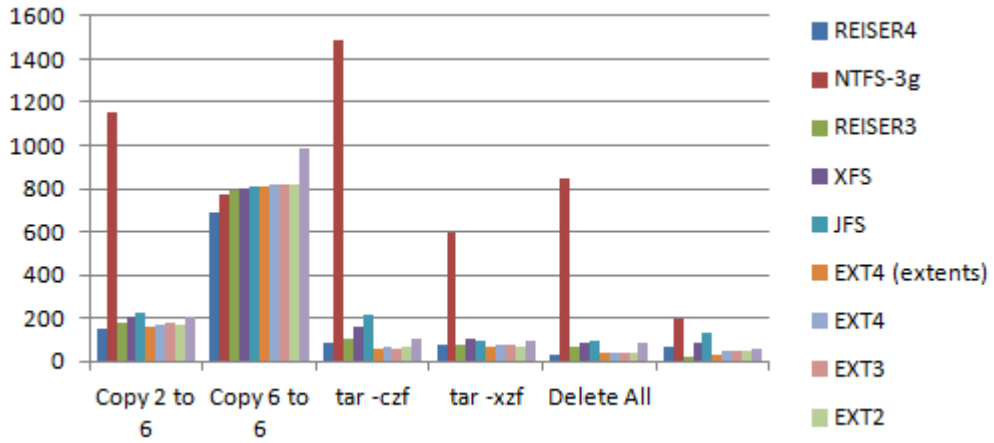
6.3.2 Nákup frameworku

Softwarové indexační a vyhledávácí frameworky
Bylo hledáno mezi softwarovými indexačními a vyhledávacími frameworky, ale byly nalezeny pouze dva frameworky, které by bylo možné teoreticky implementovat do katalogizačního systému. Jedná se o frameworky MDB search engine a Extreeme search engine. MBD search engine zaujal především svojí cenou, cena je pouhých $49.98 za nejvyšší licenci. Cena frameworku Extreeme search engine je sice o řád vyšší, ale přesto je cena pořád přijatelná.

Oba frameworky jsou si velmi podobné, mezi společné vlastnosti patří:

- mohou být provozovány pouze pod operačním systémem Windows,
- neposkytují žádné API, přes které by bylo možné data indexovat. Data se indexují přes soubory, což komplikuje celý proces indexování,
- nepodporují škálovatelnost,
- velmi slabá podpora pro nastavení vyhledávaných polí,
- maximální velikost databáze je 2TB,
- problém s předáváním výsledků vyhledávání.

Jak je patrné z následujícího obrázku č. 9, tak formát NTFS je několikanásobně pomalejší než linuxové oddíly, proto není vhodné ukládat rozsáhlý index na disk zformátovaný na NTFS. Volba formátování diskového oddílu tak má výrazný podíl jak na rychlost indexace záznamů, tak především na rychlost vyhledávání.



Obrázek 9: Srovnání formátů jednotlivých partition (LinuxHelp, 2006)

Existuje způsob, který může indexovat data v prostředí Windows a indexovaná data budou ukládána na vybraný unixový oddíl. V praxi by to vypadalo tak, že

nejčastěji by se indexovaná data ukládala přímo na aplikačním serveru. Následující obrázek č. 10 ukazuje situaci, kdy by byl index uložen na speciálním serveru. Jak je z něj patrné, tak se celý systém ještě více komplikuje. Budou existovat zákazníci, kteří budou požadovat instalaci v lokální síti, ale budou nuceni si koupit licenci pro operační systém Windows.

Obrázek 10: Způsob uložení indexu z Unixového aplikačního serveru

Následující tabulka č. 9 porovnává oba indexační a vyhledávací frameworky. Porovnání shrnuje jednak funkce jednotlivých frameworku, ale hlavně je z ní patrné, jestli jsou nebo nejsou frameworky vhodné pro použtí v nově vznikajícím katalogizačním systému.

Všechny požadavky jsou rozděleny do tří kategorií, aby bylo možné jednoduchým způsobem vyloučit frameworky, které jsou v praxi nepoužitelné.

1. zásadní – bez splnění těchto požadavků není možné framework použít a neexistuje způsob, jak požadavku vyhovět jiným způsobem,
2. důležité – jedná se o požadavky, které by měly být splněny, ale systém lze bez nich provozovat,
3. vedlejší – jedná se o požadavky, které nemusí být splněny, ale jejich splnění by přineslo uživatelům větší užitek.

Tabulka 9: Tabulka zobrazující požadavky na vyhledávací frameworky (MBD Soft, 2008; Xtreeme, 2009)

Kategorie	**Důležitost**	**MDB search engine**	**Extreeme search engine**
Rychlost vyhledávání	zásadní	vyhovující	vyhovující
Rychlost indexování	zásadní	vyhovující	vyhovující
Řazení výsledků podle relevance	zásadní	podporuje	podporuje
Řazení výsledků podle abecedy (podpora UFT8)	zásadní	nepodporuje	nepodporuje
Vyhledávání s wildcards znaky (minimálně hvězdička, uvozovky)	zásadní	nepodporuje	podporuje
Vyhledávání ve všech polích	zásadní	podporuje	podporuje
Vyhledávání ve vybraných polích	zásadní	podporuje	podporuje
Nastavení vah vyhledávaným polím	zásadní	nepodporuje	velmi slabá podpora
Podpora logických operátorů AND a OR	zásadní	podporuje	podporuje
Facetování	zásadní	nepodporuje	nepodporuje
Groupovací funkce	zásadní	nepodporuje	nepodporuje
Case sensitive a insensitive vyhledávání	zásadní	pouze case insensitive	pouze case insensitive
Vyhledávání bez diakritiky i s diakritikou	důležité	vyhledává pouze s diakritikou	vyhledává pouze s diakritikou
Vyhledávání lemmatizovaných výrazů	důležité	nepodporuje	nepodporuje
Škálovatelnost	důležité	nepodporuje	nepodporuje
Zvýraznění hledaných výrazů (i lemmatizovaných)	vedlejší	pouze nelemmatizované výrazy	pouze nelemmatizované výrazy

Podle předchozí tabulky č. 9 je zřejmé, že nalezené softwarové frameworky jsou pro použití nevyhovující, protože nesplňovaly základní požadavky na indexační a vyhledávací framework. Tyto základní požadavky není možné vynechat a není možné ani vyhledávací framework nějakým způsobem rozšířit, proto tyto frameworky není možné použít.

Indexační a vyhledávácí frameworky kombinující SW a HW

Jelikož není možné použít žádný z předchozích softwarových frameworků, proto bylo hledáno ještě mezi indexačnímy a vyhledávacími servery kombinující software a hardware. Jedná se o indexační servery v rackovém provedení, které musí být fyzicky instalovány v rámci každé instance katalogizačního systému. Servery umí indexovat přímo databázi nebo data přes poskytnutá API. Tyto indexační stroje mají několik výhod, ale také nevýhod.

Jako velkou výhodu lze uvést vysokou stabilitu, spolehlivost a rychlost vyhledávání a indexování záznamů. Je to proto, že se tyto firmy na činnost indexování a vyhledávání přímo specializují. Mezi další výhody patří možnost přímého indexování databází, HTML i Officových dokumentů.

Mezi hlavní nevýhody patří nemožnost nasazení např. v Amazon Cloudu, kde je možné využívat pouze virtuální servery a na ně instalovat požadovaný software. Další velkou nevýhodou by byla instalace přímo u zákazníkaje, který by si musel koupit vlastní indexační server, nebo by cena musela být součástí prodejní ceny, což by vedlo k zbytečnému snížení zisku z tohoto prodeje. Další nevýhodou je, že se kupuje takzvaný „blackbox“, v tomto stroji není možné provádět žádné úpravy, přidávat vlastní pluginy pro vyhledávání apod.

Byly vybrány indexační servery, které jsou zaměřeny na indexování velkého množství dat.

Tabulka 10: Tabulka zobrazující srovnání jednotlivých indexačních serverů (Goebel Group, 2005)

Vlastnosti	**Thunderstone Search Appliance APP 250**	**MaxxCAT EX-5000**	**Index Engines ES-200**	**Google Search Appliance GB-7007**	**Google Search Appliance GB-9009**
Licence	na vždy	dva roky	na vždy	dva roky	dva roky
Max. dotazů/min.	2 500	12 500	3000	300	1 000
Max. dokumentů	6 milionů	neomezeno	30 milionů	10 milionů	neomezeno
Rack	1U	1-8U	1U	multi rack	multi rack
Podpora	email, fórum, telefon	email, telefon, stránky	email, telefon, online	email, fórum	email, fórum
Cena	od $10 000	$8 995	$34 500	neznámá	neznámá

U indexačních serverů od společnosti Google se nepodařila zjistit cena. Cena těchto serverů se odvíjí individuálně podle každého zákazníka zvlášť. Abychom si mohli udělat rámcovou představu o ceně, tak v následující tabulce č. 11 bude srovnán produkt u kterého je cena známá, s produktem u kterého je cena neznámá.

Tabulka 11: Tabulka srovnávající jednotlivé produkty společnosti Google (Goebel Group, 2005)

Vlastnosti	**Google Mini**	**Google Search Appliance GB-9009**
Licence	na vždy	dva roky
Max. dotazů/min.	50	1 000
Max. dokumentů	300 000	bilion
Rack	1U	multi rack
Podpora	email, fórum	email, fórum
Cena	$ 9 990	neznámá

Z následující tabulky č. 11 vyplývá, že se nejedná o malý server, ale o velký server určený pro rozsáhlé aplikace. Vše nasvědčuje tomu, že částka se bude pohybovat ve stovkách tisíc dolarů za licenci na dva roky, což je opravdu velmi vysoká částka vzhledem k funkcím, které produkt nabízí.

Pro podporu rozhodování byla vytvořena tabulka č. 12 vícekriteriální rozhodování, která porovnává vybrané indexační servery.

Tabulka 12: Tabulka vícekriteriálního rozhodování porovnávající jednotlivé indexační servery

Kritérium	**Váha**	**Thunderstone Search Appliance APP 250**	**MaxxCAT EX-5000**	**Index Engines ES-200**
Kapacita indexovaných dat	0,20	10	30	20
Rychlost vyhledávání	0,40	10	30	10
Licence	0,20	50	5	50
Cena	0,20	30	35	20
Celkem	1,00	22	26	22

Na základě předchozí tabulky č. 12 se jako optimální varianta pro současně požadavky jeví produkt MaxxCAT EX-5000, který poskytuje nejlepší možnosti co

se týká porovnání ceny a výkonu. Tato firma nabízí i slabší verzi, což má výhody například pro samostatné instalace. Nemusí se měnit API, nebo vyvíjet nové.

V případě samostatné instalace katalogizačního systému by byla vhodnější slabší verze MaxxCat SB-250. Ten lze zakoupit s licencí na dva roky za necelé $3 000, přitom má paměť pro indexovaní 240 GB indexovaných dat.

Produkty MaxxCAT podporují indexaci databází Oracle, MSSQL i MySQL.

Není také omezen žádným limitem pro počet indexovaných souborů. Cenově vychází nejlépe, ovšem s porovnáním s konkurencí nabízí pouze dvouletou licenci, po dvou letech je nutné koupit licenci znovu. Množství dotazů za minutu převyšuje několikanásobně všechny ostatní konkurenty.

6.3.3 Vlastní framework

Při vývoji vlastního frameworku pro indexování a vyhledávání by musely být splněny všechny předchozí požadavky. Problematika vyhledávání je velmi složitá a komplexní věc, při které se používají jedny z nejsložitějších algoritmů. Jen ve zcela výjimečných případech se firmy vydávají cestou vlastní implementace vyhledávacího frameworku.

Požadované znalosti v oblasti vyhledávání jsou naprosto klíčové, pro vytvoření vlastního frameworku. Většinou se touto problematikou zabývají instituce, které ji mají jako hlavní náplň své práce.

Například vývoj technologie Lucene/Solr probíhá již více než deset let, tato firma má několik sponzorů a celou řadu programátorů. Pokud bychom se chtěli alespoň přiblížit všem funkcím co tato technologie umožňuje, pak bychom se hrubým odhadem dostali na 500 až 1 000 man-day na vývoj podobného frameworku.

V následující tabulce č. 13 jsou uvedeny časové odhady uvedené v man-day pro vývoj vlastního indexačního a vyhledávacího frameworku. Jak již bylo uvedeno výše, tento framework má vysoké nároky.

6.3.4 Odhad a srovnání časových nákladů pro vybranou dílčí část

Odhady jsou stanoveny na základě zkušeností a studiem dokumentace. Zejména u Open source projektu bylo vycházeno ze znalostí technologie Lucene, která slouží jako základ pro vybranou technologii Solr. U nákupu frameworku byl časový odhad proveden na základě poskytnuté dokumentace a API.

Každá varianta může mít některé z následujících časových nároků:

- implementace vyhledávacího frameworku – jedná se o časové nároky na vývoj katalogizačního systému podle všech požadavků. Jsou v něm zahrnuty i úpravy, které framework standardně nepodporuje a které musí být implementovány zvlášť,

Tabulka 13: Tabulka zobrazující časové odhady pro tvorbu vlastního vyhledávácího frameworku

Funkce	**Počet MD**
Základ indexačního jádra	200
Transakční indexování	50
Škálovatenost	100
Vyhledávání s wildcards	50
Facetování	50
Groupovací funkce	50
Nastavení vah vyhledávaných polí	20
Řazení podle relevance a abecedy	30
Zvýraznění hledaných výrazů, včetně lemmatizovaných	5
Vlastní dotazovací jazyk	20
Celkem	575

- implementace frameworku do katalogizačního systému – tato časová náročnost ukazuje náročnost na studium frameworku a jeho použití v katalogizačním systému,
- implementace programu pro nahlížení do indexu – nahlížení do indexu nemusí být součástí indexačního a vyhledávacího frameworku, tento program slouží zejména k hledání chyb a k zlepšování vyhledávání. V některých případech může být tento program součástí frameworku a v některých případech se jedná o program vyvinutý komunitou uživatelů používající daný framework.

Tabulka 14: Tabulka zobrazující náročnosti implementace jednotlivých variant (hodnoty jsou uvedeny v man-day)

Implementace	**Open source framework**	**Nákup frameworku**	**Vlastní framework**
Implementace vyhledávacího frameworku	5	5	575
Implementace frameworku do katalogizačního systému	20	20	10
Implementace programu pro nahlížení do indexu	0	0	20
Celkem	25	25	605

Jak je z předchozí tabulky č. 14 patrné, tak časově nejnáročnějším způsobem implementace je tvorba vlastního frameworku. Celková náročnost vlastního frameworku je 595 man-day, která je oproti použití open source frameworku nebo nákupem téměř 24x vyšší.
Odhad časové náročnosti bude mít velkou váhu při volbě vhodné koncepce pro část indexování a vyhledávání.

6.3.5 Volba vhodné koncepce pro vybranou dílčí část

Jelikož jako dílčí část byla vybrána ta nejdůležitější oblast – indexování a vyhledávání informací, proto je toto rozhodnutí klíčové a nemůže být provedeno pouze na základě vícekriteriálního rozhodování, ale také na základě dosavadních odborných zkušeností a vlastní intuice.

Pro vícekriteriální rozhodování byla vybrána následující kritéria:

- pracnost implementace – pracnost implementace se skládá z pracnosti pořízení frameworku (implementace vlastního), úprava podle všech požadavků a vlastní implementace frameworku do katalogizačního systému,
- podporované funkce – označuje funkce, které framework podporuje navíc oproti požadovaným funkcím,
- licence – je dalším důležitým kritériem, protože od způsobu licencování se odvíjí konečná cena frameworku, která může být jednorázová, nebo periodická, případně způsob licence může mít vliv na licenci výsledného katalogizačního systému,
- programovací jazyk – toto kritérium nemá velkou váhu, protože na programovací jazyk Java lze napojit např. knihovny z jazyka C, C++ a dalších. S tímto máme již zkušenosti z implementace Z39.50 server. Případně lze vytvořit malý indexační a vyhledávací server, který např. přes webové služby bude poskytovat API k indexování a vyhledávání. Přesto je preferován spíše programovací jazyk Java, který je jednodušší pro implementaci než ostatní programovací jazyky,
- možnost rozvoje – kritérium určuje, jakým způsobem je možné rozvíjet další funkce frameworku, jedná se například o různé pluginy nebo vlastní úprava části frameworku,
- konfigurovatelnost frameworku – toto kritérium určuje do jaké míry je možné framework konfigurovat, například jak je možné vytvářet vyhledávaná pole, nastavovat jim váhu pro vyhledávání, různé způsoby cachování a dalších způsobů optimalizace,

- náročnost údržby – u tohoto kritéria záleží na náročnosti údržby v rámci provozu systému. Může se jednat o nějaké periodické optimalizace indexu, které budou zlepšovat rychlost vyhledávání apod,
- hledání problémů – v rámci tohoto kritéria bude řešeno, jak hledání potenciální chyby ve frameworku, tak i hledání chyby ve vlastní aplikaci, případně špatné používání frameworku.

Tabulka 15: Tabulka vícekriteriálního rozhodování pro výběr vhodného indexačního frameworku

Kritérium	**Váha**	**Open source framework**	**Nákup frameworku**	**Vlastní framework**
Pracnost implementace	0,30	9	9	3
Licence	0,15	9	4	9
Podporované funkce	0,20	8	7	4
Programovací jazyk frameworku	0,05	9	5	9
Možnost rozvoje	0,10	7	1	9
Konfigurovatelnost frameworku	0,10	8	7	9
Údržba	0,05	8	8	9
Hledádní problémů	0,05	9	4	9
Celkem	1,00	8,45	6,35	6,20

Z předchozí tabulky č. 15 vícekriteriálního hodnocení variant vyplývá, že jednoznačně nejhorší variantou je implementace vlastního frameworku, proto ani časové odhady pro implementaci nebudou dále upřesňovány.

Jednoznačné je to i u open source projektu a koupí indexačního a vyhledávacího serveru. Podle zadaných kritérií z přechozí tabulky se jako optimální jeví varianta implementace open source indexačního a vyhledávacího frameworku.

I z vlastních zkušeností se zdá technologie Solr jako téměř ideální. Existují nějaké problémy, jako například s lemmatizovanými výrazy, ale s tímto problémem bychom se setkali u všech technologií. Jelikož máme s předchůdcem této technologie (Lucene) velmi dobré zkušenosti, pak se i v této oblasti jeví tato technologie jako nejlepší pro tyto požadavky.

Dalším kritériem je subjektivní pocit a intuice. Jelikož je projekt Solr jednoduchý na implementaci, umožňuje téměř všechny naše požadavky a hlavně nejsme

svázáni žádnou omezující licencí, můžeme si do něj stahovat různé pluginy, případně jej sami upravovat. Tento produkt má navíc širokou uživatelskou základnu, takže se s největší pravděpodobností bude udržovat a dál rozvíjet.

Další výhodou Solr je, že se jedná čistě o softwarové řešení, není k němu třeba žádný speciální hardware a proto může být instalace provedena prakticky ihned a vzdáleně. Není nutné tedy instalovat fyzicky žádný hardware a instalaci tak lze provést například do Amazon cloudu.

Na základě všech předchozích výhod a nevýhod obou řešení se v současné chvíli jeví jako optimální řešení technologie pouití Open source frameworku, konkrétně technologie Solr, proto tato technologie bude implementována do katalogizačního systému.

7 Diskuze

Velkou nevýhodou současných řešení je, že většina z nich jsou desktopové aplikace založené na principu klient-server. Knihovny tak musí mít vlastní servery a o ty se musí starat. To vede k zbytečně vysokým výdajům na provoz knihoven. Navrhované řešení (v případě cloudového řešení) tedy snižuje náklady na provoz knihovny a umožňuje i provoz knihovny v případě výpadku elektrické energie, kdy je umožněno pracovat půjčovat a vracet knihy prostřednictvím mobilního zařízení.

V případě analýzy tržního potenciálu v Polsku nebyl zjištěn aktuální stav školních knihoven. Tato informace nebyla nalezena, byla nalezena pouze informace udávající celkový počet škol v Polsku, ale tato informace nemá žádný vypovídající charakter, protože více škol může sdílet jednu knihovnu. Jelikož nově vznikající systém není určen přímo pro školní knihovny (navrhovaný systém má velmi široké zaměření), tak nepřítomnost této informace není nijak závažná.

Systém se v současné době implementuje a fáze katalogizace (ukázka systému je na obrázku v příloze č. 13 a č. 14) a webového katalogu (ukázka katalogu je na obrázku v příloze č. 15) je již hotová. Návrh celého systému nezpůsobil zatím žádné komplikace a umožňuje plynulý vývoj systému. Oproti původnímu odhadu trvala implementace katalogizace o více než 40 % déle, než se předpokládalo v průběhu návrhu systému. Při implementaci webového katalogu se odložil pouze jeho začátek, jinak implementace byla dokonce o 10 % kratší, než původní odhad. Odložení začátku implementace webového katalogu bylo způsobeno prodloužením fáze implementace katalogizace.

Nepřesný odhad pro vývoj části katalogizace byl způsoben několika faktory. Prvním z těchto problémů byla volba technologií a to hned dvou Ujorm a GWT.

V době výběru tohoto GWT frameworku byla tato technologie velkou novinkou a na diskuzních fórech se uváděl velmi jednoduchý a rychlý vývoj v tomto frameworku. Vývoj je jednoduchý a rychlý pouze v některých případech – například na tvorbu administrátorského rozhraní – jednoduchých CRUD apod. Na složitých formulářích (kde se přenáší tisíce objektů) probíhá vykreslení ve vývojovém prostředí velmi dlouho. Toto byla pouze část problému, další částí byla neznalost tohoto frameworku, bylo také nutno změnit způsob myšlení, protože tento framework používá asynchronní volání metod z klienta na server, což je pro klasicky smýšlející programátory ze začátku problém.

Technologie Ujorm je také nově vznikající databázový framework, který byl použit poprvé v komerční aplikaci. Tento framework byl použit proto, že velmi vyhovoval technologii GWT, která byla preferována pro vytvoření klientské části systému. Byly požadavky na podporu databází Oracle, MSSQL a MySQL. Odladění a opravy chyb tohoto frameworku zejména pro databázi MSSQL zabralo největší

množství času.

Při výuce databázových systémů se berou knihovní jako exemplární případy, na kterých se prezentují relační databáze. Reálná struktura databáze, která se používá v knihovních (katalogizačních) systémech je řádově složitější, než jak je nastíněno při výuce. Oblast knihovních systémů je velmi složitá a má své specifické rysy, proto je výrobců těchto systémů velmi málo.

Vývoj webového katalogu probíhal podle plánu, protože se žádné neznámé technologie nepoužily a také proto, že velmi podobný webový katalog byl implementován pro předchůdce tohoto systému – pro systém Clavius. Implementace byla sice v jiné technologie, ale nejdůležitější bylo získání znalosti a potřeb pro webový katalog z předchozí implementace.

Časové odhady pro jednotlivé funkce, které se budou implementovat, se musely prodloužit na základě předchozích zkušeností. Prodloužení doby vývoje zbytku systému nebylo o celých 40 %, ale pouze o 15 %. Je to z důvodu vyřešení největších problémů s použitými technologiemi. Tento nárůst v sobě obsahuje zpomalení vývoje v důsledku zvolených technologií a také zvýšení časových nároků na tvorbu obecného katalogizačního systému.

Navrhované řešení je navrhováno zejména pro malé a střední knihovny, které by měly být provozovány v rámci jedné instance systému. Systém je navrhovaný tak, aby byl implementován velmi obecně a tím umožnit konfiguraci pro jednotlivé knihovny. Systém dodržuje knihovní standardy jako pro ukládání a sdílení záznamů. Díky tomu je možné provozovat i velké knihovny katalogizující desítky milionů záznamů.

8 Závěr

Na základě analýzy tržního potenciálu pro knihovní systém na trhu v ČR a trzích sousedních zemí byla prozkoumána konkurence pro potřeby nově vznikajícího systému. Pro návrh systému a volbu klíčových technologií byly shromážděny požadavky na nově vznikající systém.

Dílčí část této práce byla splněna, protože tato práce byla upltněna při návrhu nového katalogizačního systému.

Volba základu systému se i na vzdory prodloužení vývoje a tím i zvýšení nákladů na vývoj se osvědčila a v současnosti umožňuje jednodušeji zpracovat většinu požadavků.

Volba vhodného frameworku pro dílčí část indexování a vyhledávání byla jednoznačně správná, neboť implementace byla jednodušší než se očekávalo a všechny požadavky tento framework bez výhrad splnil.

Návrh nového systému splňuje všechny požadavky na obecný katalogizační systém se zaměřením na knihovny a to také díky zvoleným technologiím. Volba klíčových technologií byla součástí této diplomové práce.

I když byly ze začátku problémy s některými zvolenými technologiemi, nakonec se podařilo všechny tyto problémy překonat a díky těmto použitým technologiím je nyní systém o krok vpřed oproti konkurenci. Věřím, že i přes prodloužení doby vývoje se systém uplatní na trhu v takovém rozsahu, v jakém bylo předpokládáno.

9 Literatura

ARKILLS, B. *LDAP directories explained : An Introduction and Analysis.* 1. vydání. Boston: Addison-Wesley Professional, 2003. 432 s. ISBN 978-0-201-78792-4.

BROŽOVÁ, H., Tomáš ŠUBRT a Milan HOUŠKA. *Modely pro vícekriteriální rozhodování.* 1. vydání. Praha: Credit, 2003. 178 s. ISBN 80-213-1019-7.

BRUCKNER T., Jiří Voříšek, Alena Buchalcevová a kolektiv. *Tvorba informačních systémů : Principy, metodiky, architektury.* 1. vydání. Praha: Grada Publishing, 2012. 360 s. ISBN 978-80-247-4153-6.

BYRNE, Deborah J. *MARC MANUAL : Understanding and Using Marc Records.* 2. vydání. Westport: Libraries Unlimited Inc, 1998. 263 s. ISBN 978-1563081767.

ČUMPLOVÁ, L. *Čtenář : Měsíčník pro knihovny* (2010). [online] Soubor formátu HTML. [cit. 2012-01-21]. Dostupné z WWW: <http://ctenar.svkkl.cz/clanky/2010-roc-62/10-2010/legislativa-pro-skolni-knihovny-na-zakladnich-a-strednich-skolach-75-740.htm>.

DROBÍKOVÁ, B. *Vývoj, směřování a trendy katalogizace za poslední čtyři roky: od FRBR až po revizi AACR2R v roce 2002.* Národní knihovna: knihovnická revue (2002). [online] Soubor formátu HTML. [cit. 2012-02-12]. Dostupné z WWW: <http://knihovna.nkp.cz/Nkkr0203/0203153.html>.

DWYER, J. *Pro Web 2.0 : Application development with GWT.* 1. vydání. Berkley: Aress, 2008. 480 s. ISBN 978-1-59059-985-3.

FOWLER, M. *Destilované UML.* 3. vydání. Praha: Grada Publishing, 2009. 173 s. ISBN 978-80-247-2062-3.

GŁÓWNY URZĄD STATYSTYCZNY *Kultura W 2010 R.* (2011). [online] Soubor formátu PDF. [cit. 2012-01-27]. Dostupné z WWW: <http://www.stat.gov.pl/cps/rde/xbcr/gus/PUBL_kts_kultura_w_2010.pdf>.

GNU Operating Systems *GNU GENERAL PUBLIC LICENSE* (2007). [online] Soubor formátu HTML. [cit. 2012-02-02]. Dostupné z WWW: <http://www.gnu.org/copyleft/gpl.html>.

GNU Operating Systems *GNU LESSER GENERAL PUBLIC LICENSE* (2007). [online] Soubor formátu HTML. [cit. 2012-02-02]. Dostupné z WWW: <http://www.gnu.org/copyleft/lgpl.html>.

Goebel Group *Search Appliance Comparison Matrix* (2005). [online] Soubor formátu HTML. [cit. 2012-03-12]. Dostupné z WWW: <http://www.goebelgroup.com/sam.htm>.

GUPTA, V. *Accelerated GWT : Building Enterprise Google Web Toolkit Applications.* 1. vydání. Berkley: Apress, 2008. 312 s. ISBN 978-1-59059-975-4.

InfoLIB *Knižničný systém v Slovenskej republike* (2011). [online] Soubor formátu HTML. [cit. 2012-01-23]. Dostupné z WWW: <http://www.infolib.sk/index/podstranka.php?id=727>.

JAKUBÍKOVÁ, D. *Strategický marketing : Strategie a trendy.* 1. vydání. Praha: Grada Publishing, 2008. 272 s. ISBN 978-80-247-2690-8.

LANG, H. *Theory and practice of cost analysis.* 3. vydání. Praha: Oeconomica, 2008. 78 s. ISBN 978-80-245-1409-3.

LANius *LANius historie* (2012). [online] Soubor formátu HTML. [cit. 2012-01-14]. Dostupné z WWW: <http://www.lanius.cz/profil.htm>.

LEANDER, R. *Building Application Servers : Advances in Object Technology.* 1. vydání. Cambridge: Cambridge University Press, 2000. 415 s. ISBN 9780511894480.

LinuxHelp *Some Filesystem Benchmarks* (2006). [online] Soubor formátu HTML. [cit. 2012-01-18]. Dostupné z WWW: <http://linuxhelp.150m.com/resources/fs-benchmarks.htm>.

MBD Soft *Products / MBD Search Engine / Features* (2008). [online] Soubor formátu HTML. [cit. 2012-03-05]. Dostupné z WWW: <http://www.mbd-soft.com/mbdse_features.html>.

Národní Knihovna České Republiky *Formát MARC21: čím se liší od formátu UNIMARC* (2004). [online] Soubor formátu HTML. [cit. 2012-01-29]. Dostupné z WWW: <http://www.nkp.cz/pages/page.php3?page=fond$_{M}arc21.htm$>.NIPOS*Základní statistické cit.* $2012 - 01 - 23$].*DostupnzWWW* :<*http* : *//www.nipos* − *mk.cz/wp* − *content/uploads*/2009/03/*Statistika_kultury*_2010_*Knihovny_a_publikace_web.pdf*>.

PODESWA, H. *UML For The IT Business Analyst.* 2. vydání. Boston: Course Technology PTR, 2009. 372 s. ISBN 0-521-77849-2.

PONEC, P. *The Ujorm* (2007-2011). [online] Soubor formátu HTML. [cit. 2012-01-20]. Dostupné z WWW: <http://www.ujorm.org/>.

SCHMIDT, R. *Bibliotheken in Zahlen - Daten 2010* (2011). [online] Soubor formátu HTML. [cit. 2012-02-01]. Dostupné z WWW: <http://www.bibliotheksportal.de/bibliotheken/bibliotheken-in-deutschland/daten-und-fakten/daten-2010.html>.

STATISTIK AUSTRIA *Öffentliche Bibliotheken nach Trägerschaft 2010* (2012). [online] Soubor formátu HTML. [cit. 2012-02-22]. Dostupné z WWW: <http://www.statistik.at/web_de/statistiken/bildung_und_kultur/kultur/bibliotheken/020721.html>.

LAGOZE, Carl, Herbert Van de SOMPEL. *The Open Archives Initiative Protocol for Metadata Harvesting* (07.12.2008). [online] Soubor formátu HTML. [cit. 2012-01-27]. Dostupné z WWW: <http://www.openarchives.org/OAI/openarchivesprotocol.html>.

TICHÝ, M. *Ovládání rizika : Analýza a management.* 1. vydání. Praha: C. H. Beck, 2006. 396 s. ISBN 80-7179-415-5.

TIOBE Software *TIOBE Programming Community Index for March 2012* (2012). [online] Soubor formátu HTML. [cit. 2012-03-05]. Dostupné z WWW: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

The Apache Software Foundation *The Apache Software Foundation : Apache License, Version 2.0.* (2004). [online] Soubor formátu HTML. [cit. 2012-02-05]. Dostupné z WWW: <http://www.apache.org/licenses/LICENSE-2.0.html>.

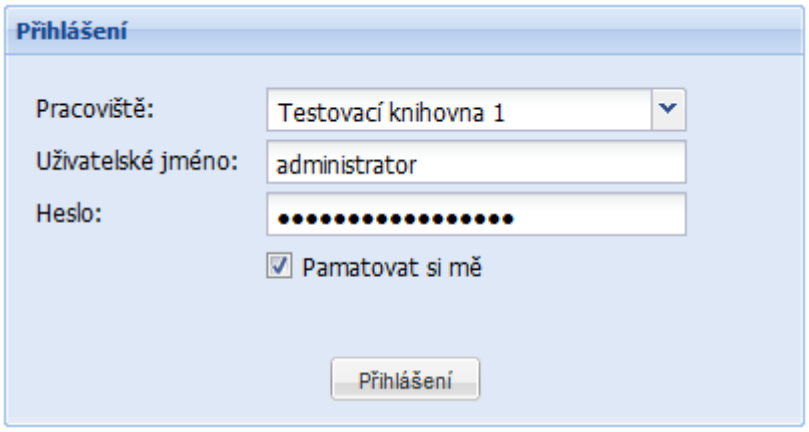
The Pragmatic Programmers *From Java to Ruby: Things Every Manager Should Know* (2006). [online] Soubor formátu PDF. [cit. 2012-03-01]. Dostupné z WWW: <http://media.pragprog.com/titles/fr_j2r/Pain.pdf>.

Webber, D. *Integrated library systems : planning, selecting, and implementing.* 1. vydání. Santa Barbara: Libraries Unlimited, 2010. 196 s. ISBN 978-1-59158-897-9.

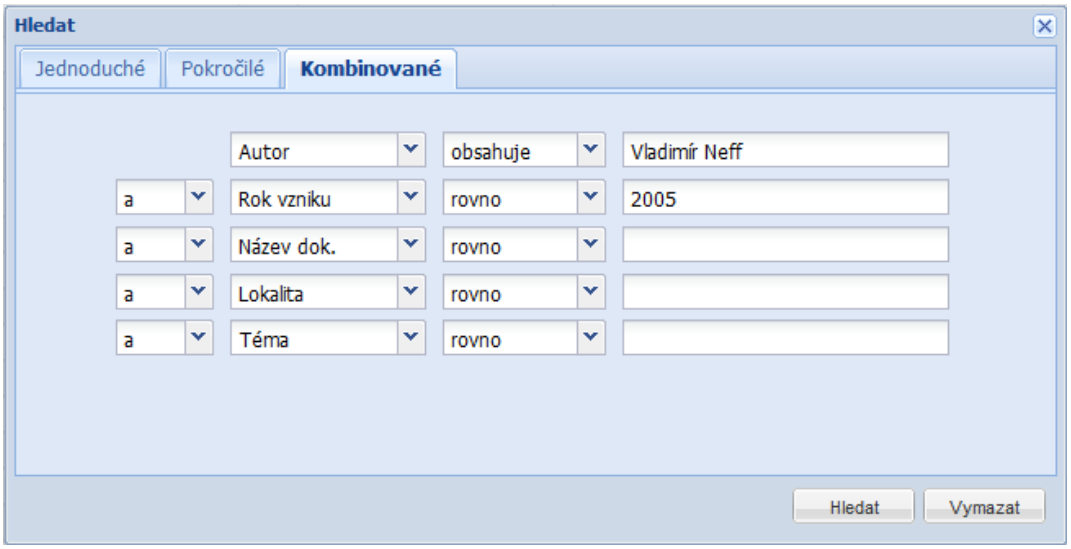
Xtreeme *Search Engine Studio Features* (2009). [online] Soubor formátu HTML. [cit. 2012-03-05]. Dostupné z WWW: <http://www.xtreeme.com/search-engine-studio/features/>.

Přílohy

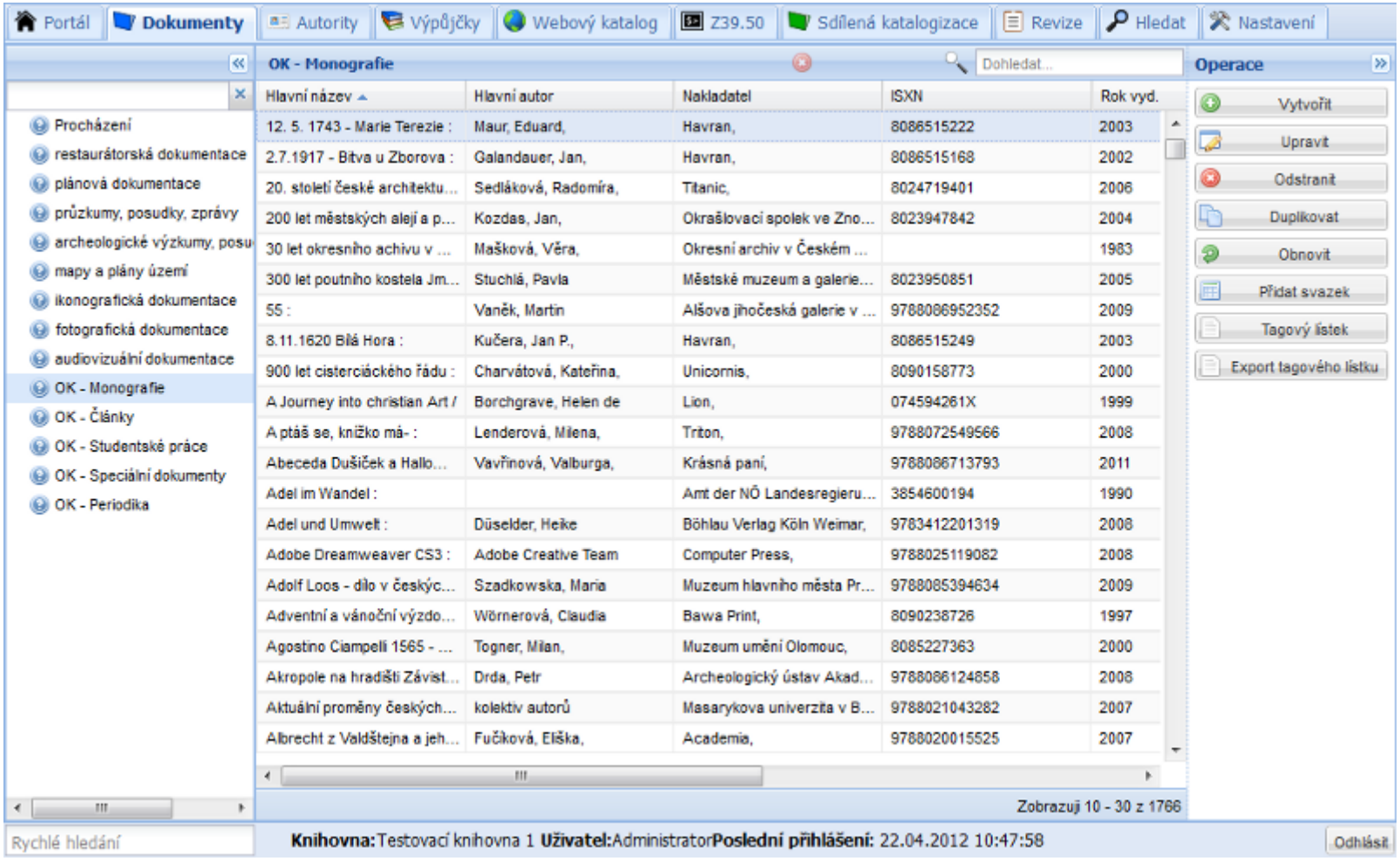
A Náhledy aplikace



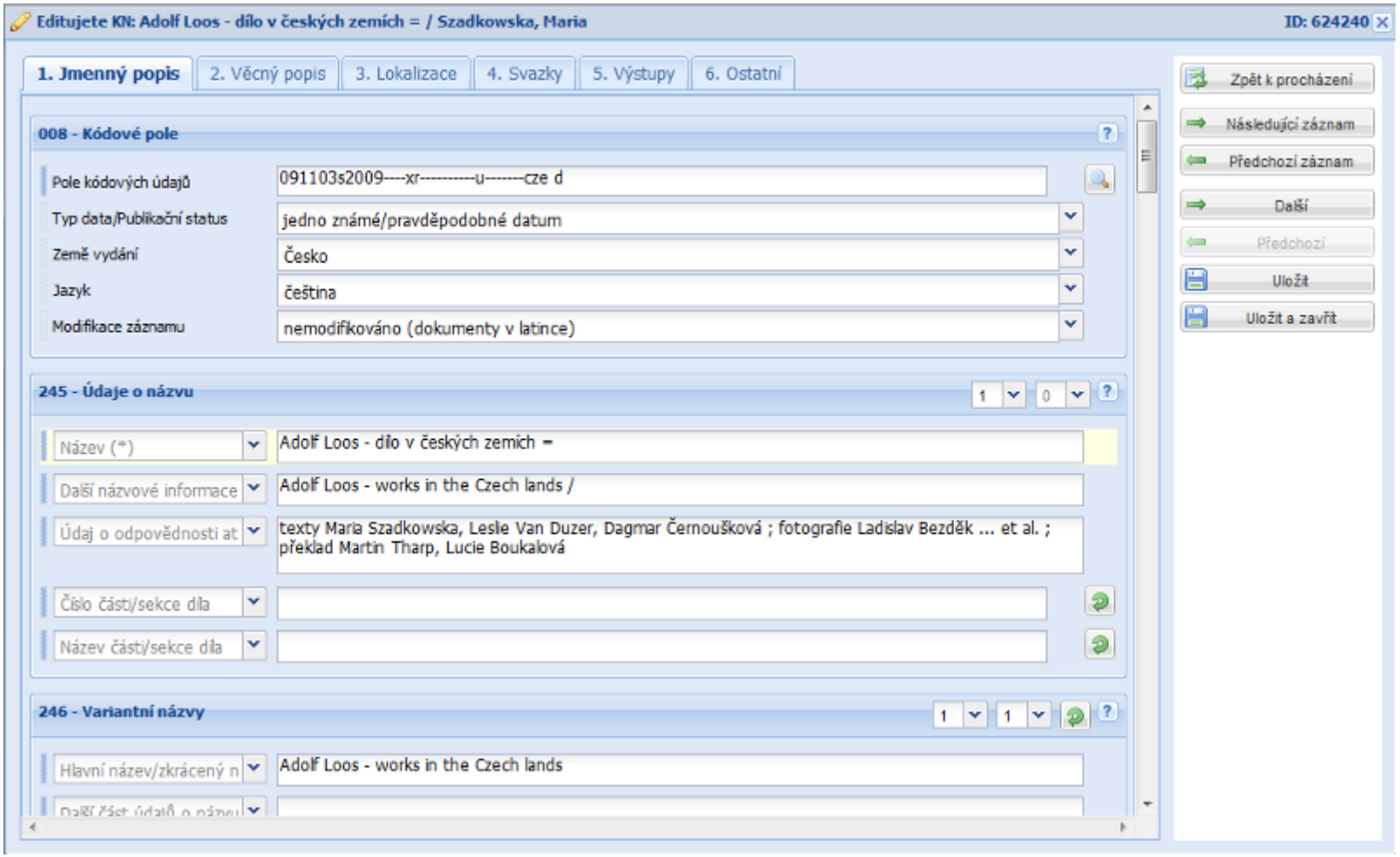
Obrázek 11: Přihlašovací formulář



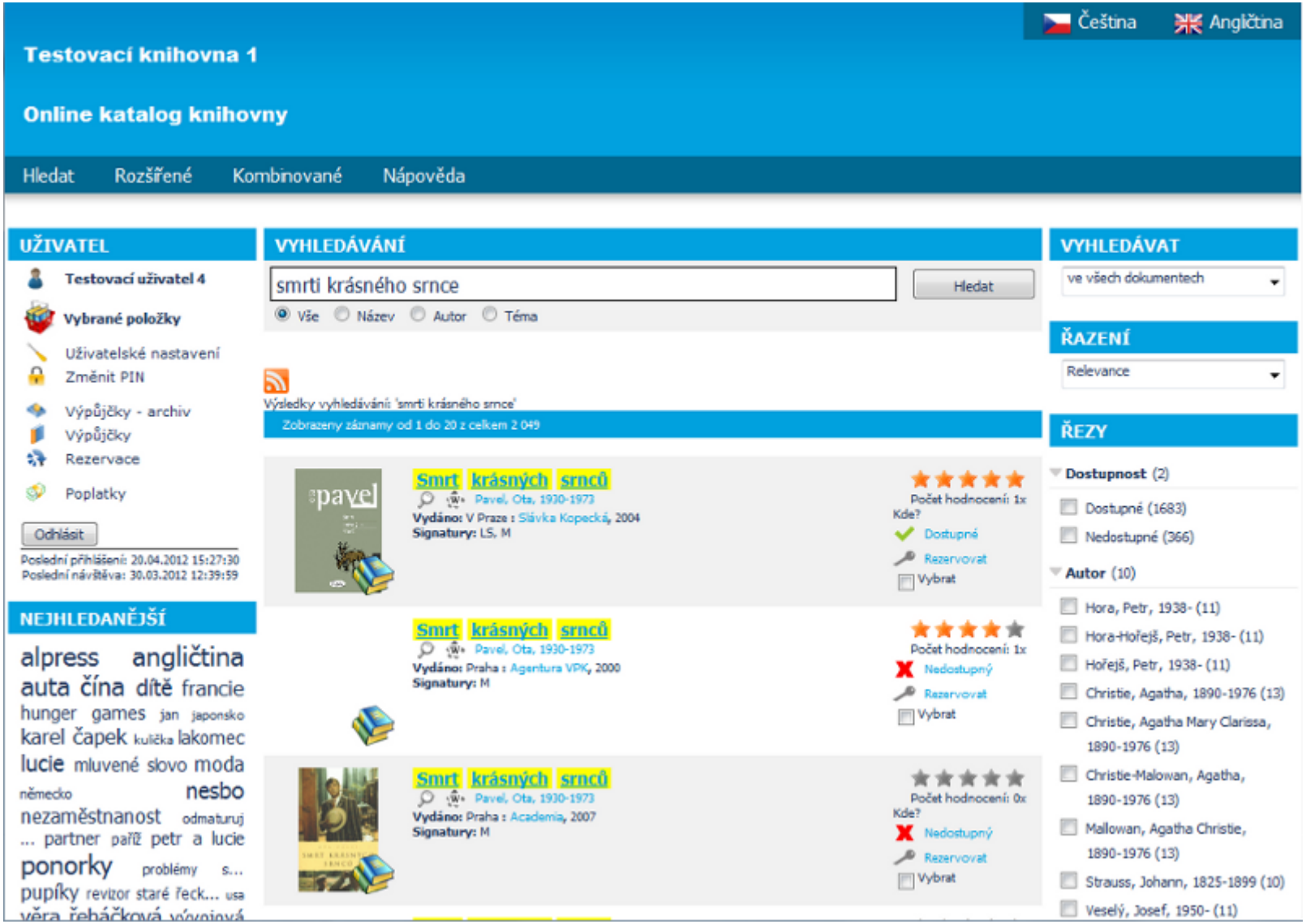
Obrázek 12: Formulář pro kombinované vyhledávání



Obrázek 13: Ukázka vlastního katalogizačního systému

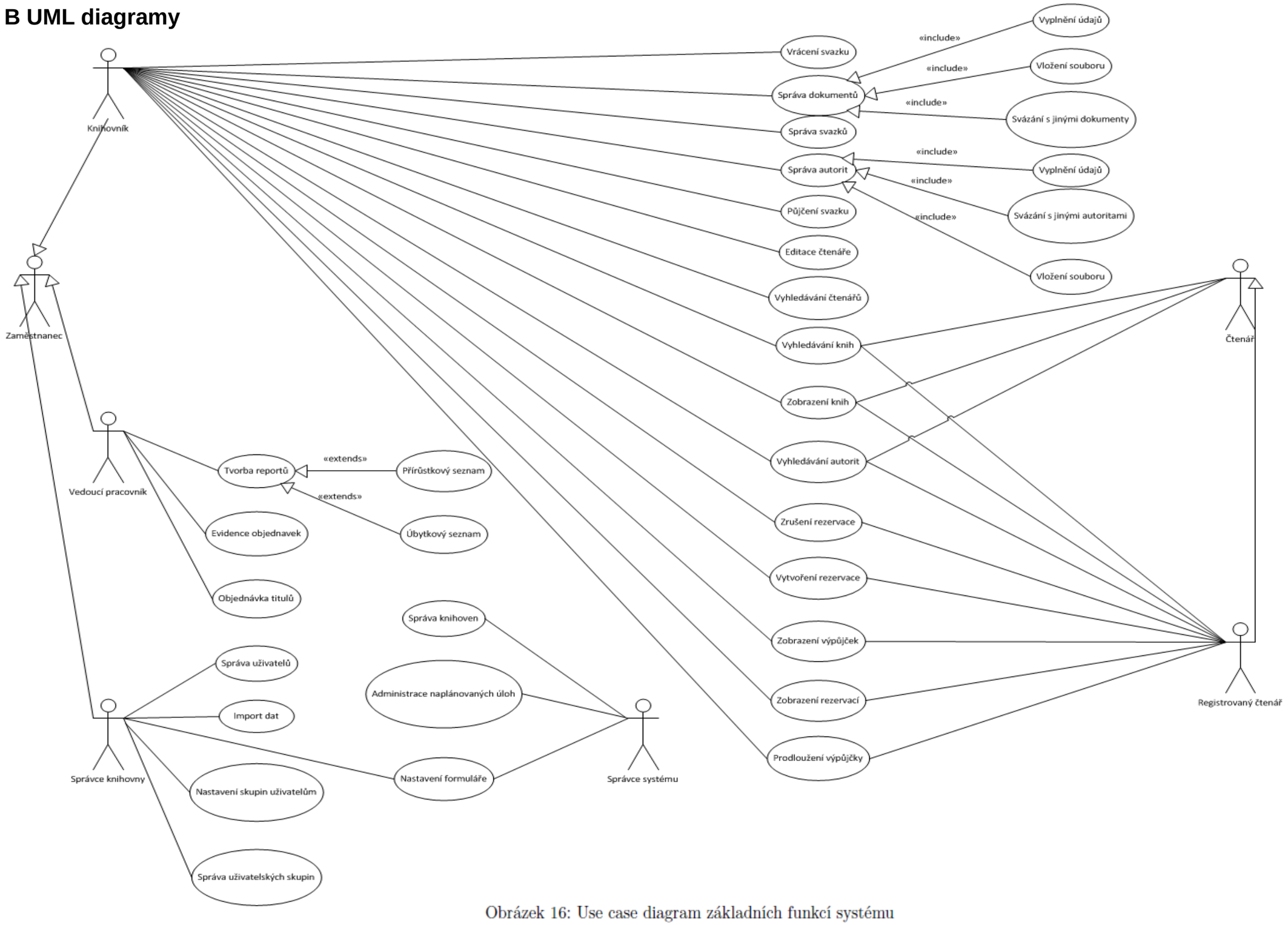


Obrázek 14: Ukázka katalogizačního formuláře

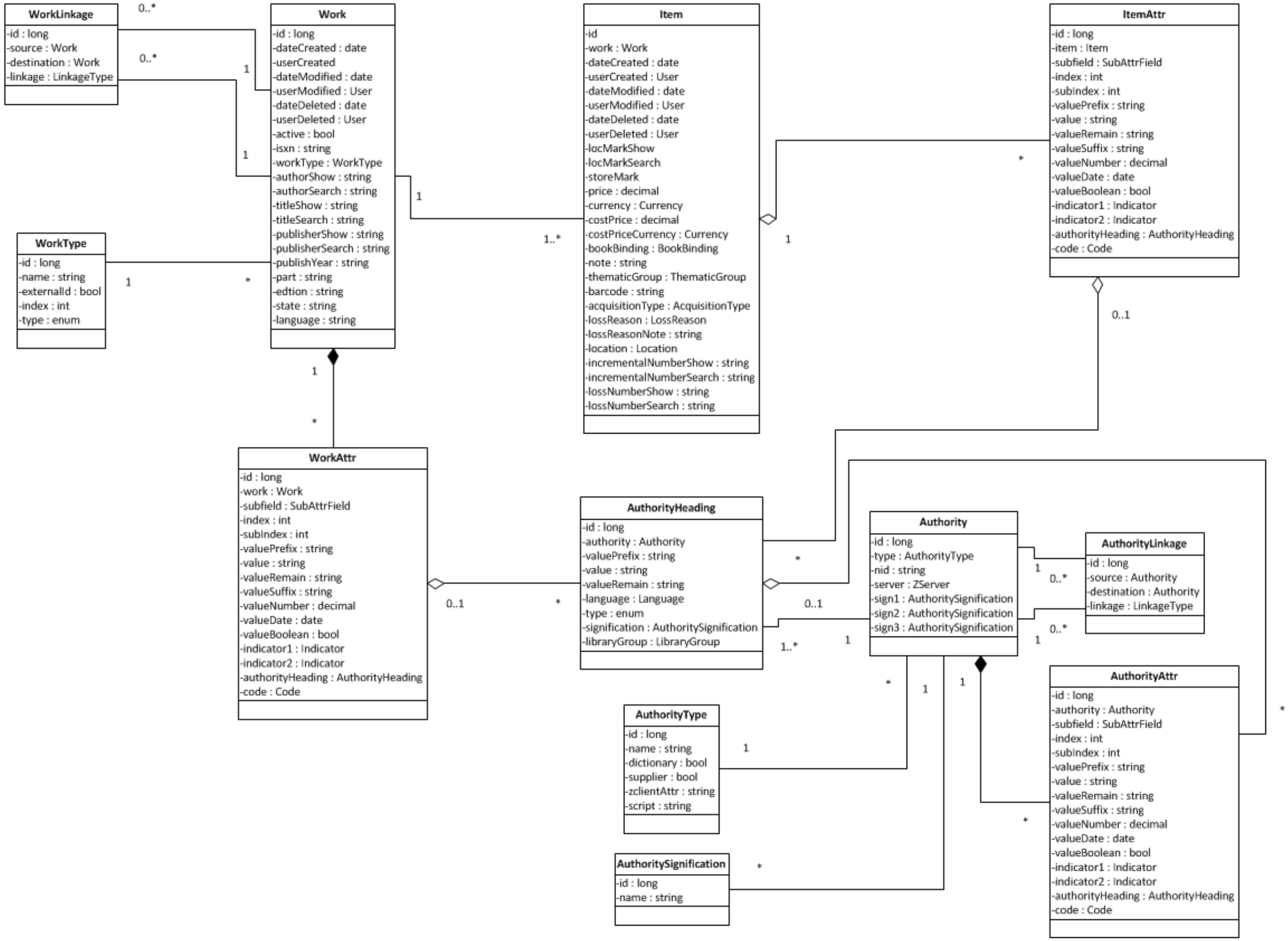


Obrázek 15: Ukázka webového katalogu

B UML diagramy



Obrázek 16: Use case diagram základních funkcí systému



Obrázek 17: Diagram tříd základu celého systému