Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Diplomová práce

Implementace protokolu NCIP do informačního systému

Vypracoval: Bc. Jan Šimeček
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2016

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan ŠIMEČEK**
Osobní číslo:	**E14737**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Implementace protokolu NCIP do informačního systému**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem diplomové práce je navrhnout a popsat implementaci protokolu NCIP do informačního systému. Protokol NCIP (NISO Circulation Interchange Protocol) je standard, který definuje protokol pro výměnu zpráv mezi počítačovými aplikacemi knihovních systémů, zpřístupňuje funkce spojené s výpůjčkami a přístupy k elektronickým zdrojům a usnadňuje spolupráci řízení těchto funkcí.

Metodický postup:

1. Studium odborné literatury.
2. Publikace výsledků rešerše.
3. Návrh, popis vývoje a implementace aplikace.
4. Zhodnocení, vypracování doporučení a závěrů.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **50 - 60 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

1. **NISO. NISO Circulation Interchange Part 1: Protocol (NCIP). Baltimore: NISO, 2015. ISBN 978-1-937522-03-2.**
2. **NISO. NISO Circulation Interchange Protocol (NCIP) Part 2: Implementation Profile 1. Baltimore: NISO, 2015. ISBN 978-1-937522-04-9.**
3. **PECINOVSKÝ, Rudolf.** *Java 8: úvod do objektové architektury pro mírně pokročilé.* **1. vyd. Praha: Grada, 2014, 655 s. Knihovna programátora (Grada). ISBN 978-80-247-4638-8.**
4. **PECINOVSKÝ, Rudolf.** *OOP: naučte se myslet a programovat objektově.* **Vyd. 1. Brno: Computer Press, 2010, 576 s. ISBN 978-80-251-2126-9.**

Vedoucí diplomové práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: **9. ledna 2015**
Termín odevzdání diplomové práce: **15. dubna 2016**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 27. března 2015

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum

4. 4. 2016

Podpis studenta

Jan Šimek

Děkuji vedoucímu diplomové práce Mgr. Radimu Remešovi
za cenné rady, připomínky a metodické vedení práce.

Anotace

Tato práce se snaží popsat možnost implementace protokolu NISO Circulation Interchange Protocol. V teoretické části bude zmíněna jeho historie, struktura a důležitost protokolu. Dále zde bude zmíněn značkovací jazyk XML a protokol HTTP. V praktické části bude vytvořena aplikace v jazyce C# komunikující pomocí protokolu NCIP. O zpracování XML se postará třída XmlSerializer. Pro testování bude využita a upravena aplikace NCIP V2 Dummy Responder, která využívá XC NCIP Toolkit v jazyce Java.

OBSAH

ÚVOD

Jako v každém IT oboru i v knihovních systémech je třeba, dodržovat určité protokoly a standarty, aby byly tyto systémy konkurence schopné. Tato diplomová práce se zaměřuje konkrétně na protokol NISO Circulation Interchange Protocol (NCIP), který definuje výměnu zpráv mezi knihovními systémy ve formátu XML přes protokol HTTP.

V praxi protokol NCIP slouží k tomu, aby mezi sebou mohla komunikovat zařízení a systémy od různých dodavatelů knihovních systémů. Zákazníkem těchto firem jsou knihovny. Pořizování knihovního systému nemusí být pevně svázáno s nákupem knihovního příslušenství, jako je např.: samoobslužný pult self-check. Knihovna si může pořídit nejdříve knihovní sytém a teprve později vybírat self-check, aniž by musela zvolit stejného výrobce. Tato situace může nastat, pokud knihovna vypíše dvě oddělená výběrová řízení. První na dodání knihovního systému a druhé na příslušenství k němu. Tímto může dojít k tomu, že dodavatelé budou různí. Následně nemusí uspět firma, která nabízí zařízení/software nepodporující protokol pro komunikaci s již nakoupeným systémem. Z toho důvodu je důležité, aby např.: self-check podporoval komunikaci přes NCIP protokol. Podobná situace vznikne, jestliže firma, která dodala jednu z částí, již nebude existovat. V tom případě bude třeba, aby nový software bez potřebných úprav byl schopný komunikovat s již používaným řešením.

Cílem této práce je, popsat možnosti implementace protokolu NCIP. Bude vybrán jeden způsob a dojde k vytvoření klientské aplikace (Initiator), která musí být schopná komunikovat s druhou aplikací (Responder) pomocí protokolu NCIP.

Práce se skládá ze dvou částí. Teoretická část začíná popisem základních pojmů. Dále následuje popis metodiky, podle které se postupovalo během tvorby této práce. Největší část je věnována samotnému protokolu NCIP. Je zde zmíněna jeho historie a organizace, která stojí za jeho vznikem. Větší pozornost je věnována struktuře a principu fungování jako např.: povinné a nepovinné elementy nejdůležitějších služeb. Pro pochopení protokolu je zde nastíněn i základní popis jazyku XML a komunikace přes protokol HTTP.

Praktická část začíná analýzou možností implementace protokolu. Initiator byl vytvořen v jazyce C#. Následně bylo vybráno nejvhodnější řešení pro vytvoření serverové aplikace (Responder). V poslední části je komunikace mezi aplikacemi podrobena testování. Praktická část se tedy skládá především z nejdůležitějších částí zdrojového kódu použitého

při tvorbě těchto aplikací. V závěru práce jsou shrnuty výsledky testování komunikace aplikací, jsou vyhodnoceny cíle diplomové práce a navrženo další možné rozvinutí dané problematiky.

1 DEFINICE POJMŮ

Tabulka 1: Definice pojmů

Pojmy	Definice
Metoda	Je nositelem dovednosti objektu.
Třída (class)	Předpis pro vytvoření objektu.
Objekt	Konkrétní instance třídy.
GUI	Je grafické uživatelské rozhraní.
Open Source	Jde o volně šiřitelný software, kterému lze upravovat zdrojový kód.
Agency	Organizace, která půjčuje Item nebo poskytuje jiné služby pro User. Vztah mezi Agency a User, Agency a Item je vidět z faktu, že identifikátor Agency je obsažen v Item i User. Agency je např.: knihovna.
Item	Je chápán jako fyzický předmět či elektronická informace patřící do kolekce Agency. Jsou to např.: svazky.
Object Class	Předpis instance objektu. Standard definuje tři třídy: Agency Object, Item Object a User Object.
Object Instance	Konkrétní instance třídy určitého objektu. Object Instance může být vytvořen s konkrétními daty, funkcemi nebo vztahy.
Protocol	Sada formálních pravidel, kterými se řídí přenos dat přes síť. Protokoly aplikace řeší formátování dat, včetně syntaxe zpráv: dialog mezi zahajující aplikací a odpovídající aplikací, znakové sady, sekvencování zpráv, atd.
Service	Dvojice zpráv, která se skládá z iniciace a reakce na zprávu, která poskytuje specifické funkce, s ohledem na výměnu dat týkajících se běžných činností.
User	Uživatel či organizace, která bude používat předmět nebo službu poskytovanou agenturou. V protokolu to bývá nejčastěji čtenář.

Zdroj: vlastní zpracování

2 METODIKA

Při tvorbě této práce se postupovalo podle níže uvedených bodů:

1. **Studium teorie**
   K pochopení dané problematiky je třeba nastudovat teorii k danému tématu.
2. **Porovnání a analýza možností implementace protokolu**
   Před samotným návrhem aplikace je důležité, prozkoumat možnosti implementace protokolu.
3. **Návrh aplikace**
   Po analýze následuje návrh klientské aplikace, která bude schopná komunikovat přes NCIP protokol.
4. **Implementace aplikace**
   S hotovým návrhem je možné provést implementaci konkrétní aplikace.
5. **Test aplikace**
   Hotová aplikace bude podrobena testování.
6. **Zhodnocení výsledků v závěru práce**
   V závěru práce budou vyhodnoceny výsledky a předem stanovené cíle práce.

3 NISO Circulation Interchange Protocol

3.1 National Information Standarts Organization

Protokol NCIP vyvinula organizace National Information Standarts Organization (NISO), která je americkou neziskovou organizací vyvíjející technické standarty určené pro knihovní systémy. Organizace byla založena v roce 1939. Organizace spojuje tradiční a nové technologie, aby vyhověly kompletním požadavkům, jako je načítání nebo uchování dat. (NISO, 2015b)

Organizace je podporována přímo uživateli těchto služeb např.: knihovnami. Během let NISO uspořádalo mnoho setkání a workshopů, jejichž výsledkem byly nové standarty. Dále se organizace snaží, aby standarty odrážely globální potřeby. (NISO, 2015b)

3.2 Protokoly v IT

Protokol je v IT chápán jako předpis pro vzájemnou komunikaci. Buď mezi hardwarem, nebo softwarem. Jsou to tedy určitá pravidla pro výměnu zpráv mezi dvěma subjekty. Protokoly usnadňují vývoj počítačových programů.

Dostupnost specifikace protokolu napomáhá k rychlému rozšiřování počítačových technologií. Tím vzniká možnost, aby spolu mohla komunikovat dvě zařízení, která mají odlišný účel, aniž by to museli tvůrci předpovídat.

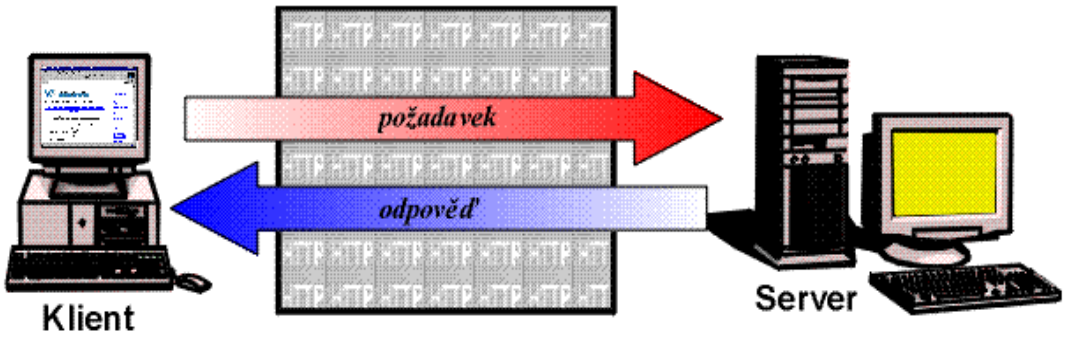
Dnes mají největší význam internetové protokoly, např.: TCP/IP, HTTP nebo POP3.

3.3 Hypertext Transfer Protocol

Nejpoužívanější službou internetu je WWW (World Wide Web). K datům se zde přistupuje pomocí URI. URI je řetězec, který vyjadřuje umístění dokumentu. Komunikace probíhá právě pomocí HTTP. HTTP je aplikační protokol, který zprostředkovává komunikaci mezi webovým prohlížečem a serverem. Přitom využívá síťový protokol TCP/IP. Pro šifrování lze použít např.: SSL, který je složkou mezi HTTP a TCP/IP. (Novák, 2006, s. 9)

Protokol je založen na způsobu komunikace: požadavek-odpověď. Klient (webový prohlížeč) vytvoří spojení a odešle požadavek, na který server odpoví, viz obrázek 1.

Obrázek 1: Komunikace přes protokol HTTP



Zdroj: (Kosek, 2000c)

3.4 eXtensible Markup Language

XML je značkovací jazyk pro výměnu informací prostřednictvím World Wide Web (WWW). Tvůrcem tohoto jazyku je W3C (World Wide Web Consortium). XML byl vytvořen na základě zkušeností s předchozími značkovacími jazyky, např.: HTML. (Bradley, 2000, s. 18)

3.4.1 Historie XML

Od počátku je jednou z hlavních funkcí počítačů příprava a publikování textu. V 60. letech výrobci osvitových jednotek používali vlastní jazyky pro jejich ovládání. Do dokumentů se vkládaly speciální řídící sekvence, které umožňovaly ovládat konkrétní osvitovou jednotku. To vedlo k tomu, že takto vytvořený dokument byl použitelný pouze pro přístroje jednoho výrobce. V té době si o univerzálnosti formátu PDF mohli nechat pouze zdát. Proto vznikly konvertory, které umožňovaly obecné příkazy převést do jazyka konkrétního zařízení. Nejrozšířenějšími z nich byly troff a TEX. I dnes je TEX v možnostech formátování nepřekonatelný. (Kosek 2000b, s. 11 – 12)

Dnes je potřeba stejné informace prezentovat v mnoha podobách. Proto bylo zapotřebí znát logickou strukturu těchto informací. Byl tedy vytvořen jazyk, který neoznačuje vzhled textu, ale význam jednotlivých částí. Těmto jazykům říkáme značkovací jazyky neboli markup languages. (Kosek 2000b, s. 12)

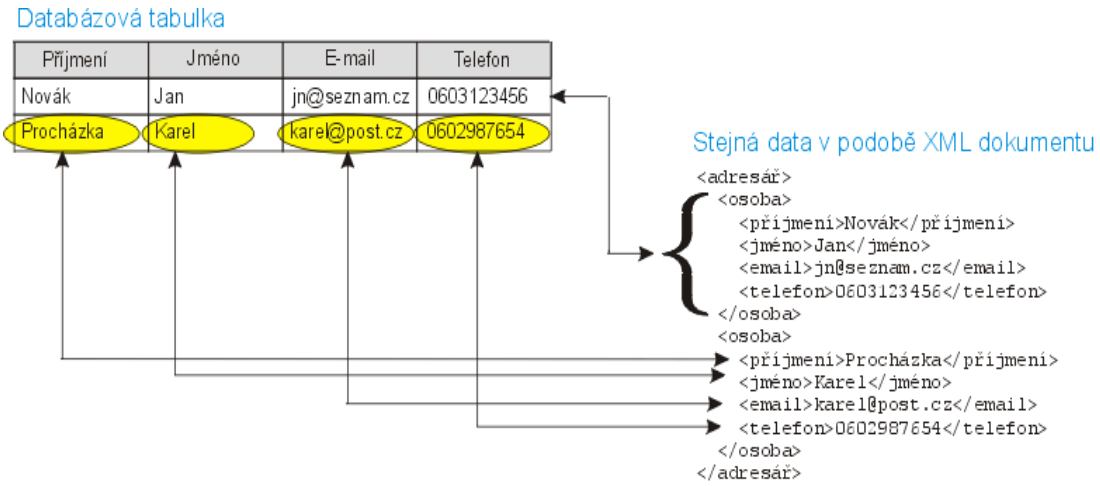
K nejstarším značkovacím jazykům patří např.: GML (Generalized Markup Language). Jazyk vytvořili páni Charles Goldfarb, Edward Mosher a Raymond Lorie. Cílem tohoto jazyku bylo zpracování textů pro IBM. V 80. letech se potvrdil potenciál GML a na jeho základě začala organizace ANSI vyvíjet jazyk, který umožnil uživateli nadefinovat si

vlastní sadu značek, která byla nejvhodnější pro konkrétní druh dokumentu. Sdružení GCA vytvořilo jazyk GenCode. Cíle obou organizací byly stejné, a proto se spojily, čímž vznikl SGML (Standard Generalized Markup Language). Tento jazyk byl definován v normě ISO 8879 v roce 1986. (Kosek 2000b, s. 13)

V současné době nejznámější aplikací SGML je právě jazyk HTML (Hypertext Markup Language), který se hojně používá pro tvorbu webových stránek. Problém spočíval v tom, že jazyk SGML byl až příliš komplexní, to znamená, že se v praxi používal jen jeho zlomek. Na druhou stranu se ukázalo, že pevně daná skupina značek u jazyku HTML nestačí. Tyto dva problémy daly vzniknout právě jazyku XML. Na rozdíl od SGML jsou některé parametry již předem určeny např.: použité oddělovače. Oproti SGML je syntaxe přísná, takže je s ním snazší práce. (Young, 2002, s. 36 – 37)

Pomocí značek v dokumentu není problém do jazyku XML zachytit i obsah tabulky z relační databáze viz obrázek 2 níže:

Obrázek 2: XML není určeno jen pro texty, poradí si i s databázovými daty.



Zdroj: (Kosek, 2000b)

3.4.2 Syntaxe XML

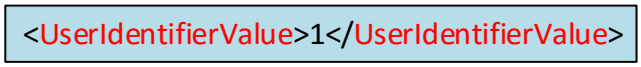
V současné době je bezesporu důležitá komunikace mezi různými softwary. Není proto možné, aby každý výrobce používal své vlastní formáty. Zde se dostáváme k NCIP. NCIP je tedy především předpis struktury pro zprávy, které si mezi sebou knihovní systémy vyměňují pomocí jazyku XML.

XML dokument se skládá z elementů, které jsou do sebe vnořené. Úplně nahoře obsahu dokumentu musí být pouze jeden element. U protokolu NCIP je to NCIPMessage. Jeden

element může obsahovat neomezené množství dalších elementů. Je to způsobeno tím, že se některé informace ve zprávě opakují – je tedy potřeba opakovat i elementy.

Elementy se označují pomocí tzv. tagů. Většinou se elementy skládají z počátečního a ukončovacího. (Kosek 2000a, s. 24)

Obrázek 3: Element UserIdentifierValue



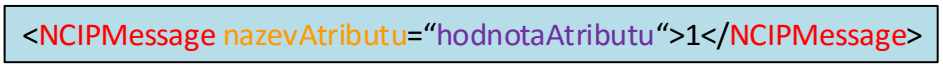
Zdroj: vlastní zpracování

Na obrázku 3 vidíme element UserIdentifierValue, který je ohraničený počátečním tagem <UserIdentifierValue> a ukončovacím tagem </UserUdentifierValue>. Z obrázku 3 vyplývá, že elementy se zapisují mezi znaky: větší < a menší >. Přičemž ukončovací tag se odlišuje od počátečního pomocí „/“. V tomto příkladu má element UserIdentifierValue hodnotu 1.

Pokud element nemá žádnou hodnotu, tak lze použít pouze jeden tag, tedy <UserUdentifierValue/>. Takovéto tagy pak nesou pouze pravdivostní hodnotu. Jeho přítomnost pak znamená hodnotu true.

Základem každého dokumentu jsou tedy elementy. Každý element může obsahovat ještě atributy. Tyto atributy se používají k upřesnění významu elementu.

Obrázek 4: Atribut elementu



Zdroj: vlastní zpracování

Na obrázku 4 vidíme, že atributy se zapisují do závorek spolu s počátečním tagem elementu (zvýrazněno žlutým písmem). Hodnota atributu se zapisuje za rovná se do uvozovek nebo do závorek (zvýrazněno fialovým písmem). Více atributů u jednoho elementu se odděluje mezerou.

Tabulka 2: Znaky, které nelze zapsat do obsahu.

Znak	Nahrazení
<	&lt;
>	&gt;
&	&amp;
uvozovky	&quot;
apostrofy	&apos;

Zdroj: vlastní zpracování

V tabulce 2 jsou ve sloupci Znak znaky, které z důvodu syntaxe XML nelze použít v obsahu elementů dokumentu. Nahrazují se hodnotou uvedenou v sloupci Nahrazení.

Obrázek 5: XML deklarace

```
<?xml version="1.0" encoding="UTF-8"?>
```

Zdroj: vlastní zpracování

Na obrázku 5 je zobrazena XML deklarace, kterou začíná každý XML dokument. Tato deklarace obsahuje použitou verzi XML (žlutě zvýrazněné) a použité kódování (fialové zvýraznění).

3.5 NISO Circulation Interchange Protocol

Protokol NISO Circulation Interchange Protocol (NCIP) známý také pod názvem Z39.83 byl představen v roce 2002. V roce 2008 byl předělán na verzi 2.0, což přineslo větší rozšiřitelnost, lepší servis a zpracování chyb. V roce 2011 přišla verze 2.01 a nakonec v roce 2012 byla vydána verze 2.02. (NISO, 2015a)

Standart definuje soubor objektů a služeb, které mezi sebou komunikují pomocí zpráv. NCIP slouží k usnadnění automatizace úkolů, výměně dat a poskytuje informace pracovníkům knihovny. Každá služba se skládá z žádosti od zahajující aplikace (Initiator) a z reakce odpovídající aplikace (Responder). Je možné, aby obě úlohy hrál jeden samostatný software, který by požadoval a zároveň přijímal. Nicméně většinou jsou zapojeny alespoň dvě aplikace. Původní vize byla pokrýt tři hlavní oblasti využití:

a) přímé konsorciální výpůjčky (DCB)

b) oběh a meziknihovní výpůjční interakce (C-ILL)
c) samoobslužný oběh (NCIP STANDING COMMITTE, 2011)

Postupem času NCIP Standing Committee (dříve znán jako NCIP Implementers Group) si uvědomil, že obecný pojem "sdílení zdrojů" zahrnuje pracovní postupy spojené jak s DCB tak s C-ILL. (NCIP STANDING COMMITTE, 2011)

V rámci sdílení zdrojů může být iniciátorem aplikace používána zaměstnanci knihovny pro požádání o položku ze vzdáleného systému, která není v místím systému k dispozici. NCIP může sloužit ke zjištění, zda je ve vzdáleném systému položka k dispozici a pokud ano, aby byla odeslána do místa dotazu. (NCIP STANDING COMMITTE, 2011)

Pro samoobslužné služby může být iniciátorem samoobslužný terminál (self-check), který poslouží uživatelům pro kontrolu a případné vypůjčení vlastních položek. (NCIP STANDING COMMITTE, 2011)

3.5.1 Struktura NCIP

3.5.1.1 Typy služeb

Již bylo zmíněno, že každá služba se skládá ze zprávy a odpovědi. Standart rozděluje tyto služby na tři základní typy:

a) Lookup Service Type
b) Update Service Type
c) Notification Service Type

Jejich výčet je možné vidět v příloze 1.

3.5.1.2 Lookup Service Type

Lookup Service Type obsahuje služby, které dovolují zahajující aplikaci zeptat se druhé aplikace na data o instanci objektu. Aplikace, která odpovídá, musí odeslat zprávu zahajující aplikaci. Buď odešle požadovaná data, nebo zamítne jejich vyhledávání. V některých případech nemusí odpovídající aplikace poskytnout požadovaná data. Například pokud jsou data nedostupná nebo pokud je k nim omezený přístup. Nicméně pokud je Problem element vrácen v odpovědi, nejsou vráceny ani nepovinné údaje ani požadované. Každá služba nebo odpověď se skládá ze tří možných údajů. Těmi jsou údaje povinné, nepovinné popřípadě odpověď může obsahovat i vyžadované údaje zahajující aplikace. Údaje nejsou seřazené podle abecedy, ale podle pořadí, jak následují za sebou v protokolu. (NISO, 2012d, s. 17)

Jako příklad je zde uvedena nejzákladnější služba Lookup User. Tato služba i s ostatními základními službami je znázorněna v příloze 2.

3.5.1.3 Update Service Type

Update Service Type zahrnuje služby, které dotazující se aplikaci umožňují požádat odpovídající aplikaci o změnu (přidání, smazání atd.) dat o Agency, Item nebo User. Tyto služby můžou obsahovat transakce k vytvoření/upravení vztahů mezi Object Instance. Odpovídající aplikace vrací zprávu o přijetí požadavku nebo o jeho odmítnutí. V některých případech nemusí odpovídající aplikace poskytnout požadovaná data. Například pokud jsou data nedostupná nebo pokud je k nim omezený přístup. Nicméně pokud je Problem element vrácen v odpovědi, nejsou vráceny ani nepovinné údaje ani požadované. Odpověď na tyto transakce by měla o tom obsahovat data. (NISO, 2012d, s. 24)

Jako příklad je uvedena služba Renew Item, která je znázorněna v příloze 3.

3.5.1.4 Notification Service Type

Notification Service Type umožňuje zahajující aplikaci oznámit druhé aplikaci, že nastala změna či vytvoření u User, Item nebo Agency. Tyto služby mohou zahrnovat oznámení o transakcích, které vytvořily/upravily spojení mezi Object Instance. Odpovídající aplikace by měla v odpovídající zprávě poslat potvrzení. Protože se jedná o jednostranné potvrzení, může odpovídající aplikace pouze potvrdit nebo oznámit, že nerozuměla. (NISO, 2012d, s. 41)

3.5.1.5 Object Classes a Object Definitions

Standart specifikuje tři hlavní Object Classes:

a) Agency Object
b) Item Object
c) User Object (NISO, 2012d, s. 14)

Rozdíl mezi např.: Item a Item Object je tedy v tom, že Item je konkrétní instance třídy Item Object.

3.5.1.6 Agency Object

Agency Object představuje knihovnu nebo organizaci, která půjčuje předměty (Items) z její kolekce nebo poskytuje služby jednomu či více uživatelům (User). Agency Object je složen z následujících elementů:

a) Agency Address Information

b) Agency User Privilege Type
c) Application Profile Supported Type
d) Authentication Prompt
e) Consortium Agreement
f) Organization Name Information (NISO, 2012d, s. 14)

3.5.1.7 Item Object

Item Object představuje fyzickou nebo elektronickou entitu, která je součástí kolekce Organizace (Agency), která může být půjčována nebo poskytována uživateli (User). Může to být cokoliv od svazků až po stolní hry. Item Object je složen z následujících elementů:

a) Bibliographic Description
b) Circulation Status
c) Electronic Resource
d) Hold Queue Length
e) Item Description
f) Item Use Restriction Type
g) Location
h) Physical Condition
i) Security Marker
j) Sensitization Flag (NISO, 2012d, s. 14)

3.5.1.8 User Object

User Object představuje osobu nebo organizaci, která využívá služeb organizace (Agency). Nejčastěji to jsou čtenáři knihovny. User Object je složen z následujících elementů:

a) Authentication Input
b) Block Or Trap
c) Date Of Birth
d) Name Information
e) Previous User Id(s)
f) User Address Information
g) User Language
h) User Privilege
i) User Id (NISO, 2012d, s. 15)

3.5.2 Nejdůležitější služby

Ačkoli standart definuje velké množství služeb, nejzákladnějších služeb je právě pět. Jsou to:

j) Check In Item
k) Check Out Item
l) Lookup Item
m) Lookup User
n) Renew Item (NCIP STANDING COMMITTE, 2011)

4 Návrh a implementace řešení

V této části diplomové práce jsou rozebrány možnosti implementace NCIP protokolu do informačního systému. Existují tři možnosti, jak implementaci provést:

1. Skládat řetězce dohromady
2. Využít XML toolkit
3. XC NCIP Toolkit

První možnost v současné době není správným řešením, protože skládání řetězců dohromady není objektové a tudíž zastaralé a nepraktické.

Druhou možností je využití některé XML toolkitu, který poskytne dobře objektově orientované zpracování elementů pro zprávu.

Poslední možností je využití XC NCIP Toolkit, který je přímo přizpůsoben pro práci s NCIP protokolem. Je licence MIT/X11 a lze ho stáhnout v jazyce JAVA. Z příkladu je jasné, že právě druhá a třetí možnost je adekvátní pro implementaci NCIP protokolu do informačního systému.

4.1 Initiator

Initiator byl vytvořen v programovacím jazyce C# za použití Windows Presentation Foundation (WPF).

Pro Initiator je nejlepší použití druhé možnosti. Není totiž potřeba implementovat všechny služby. Jako zahajující aplikace očekává pouze odpovědi na služby, které sám odeslal. Tím programátor získá volnost v psaní svého kódu. Z tohoto důvodu byla pro tvorbu XML zpráv využita třída XmlSerializer, která poskytla objektové zpracování XML.

Vše bude ukázáno na tvorbě zprávy pro službu Lookup User.

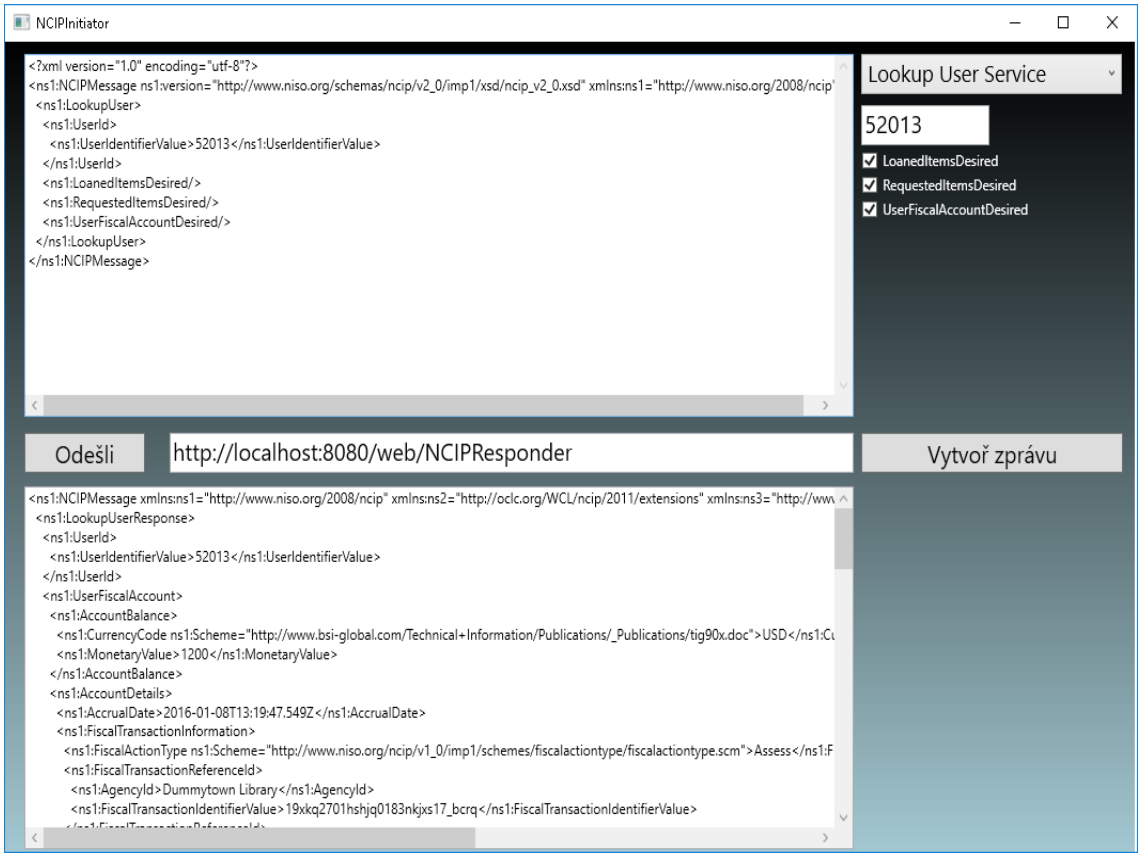
4.1.1 Třídy

Initiator se skládá z hlavní třídy MainWindow, která dědí z Window a je tedy GUI aplikace. Další třídou je ToXml, která zajišťuje tvorbu zpráv XML. Na každou službu připadá jedna třída. Každá z těchto tříd je uložena v cs souboru, který obsahuje i třídy zastupující jednotlivé elementy potřebné do zprávy služby např.: UserIdE. Pro názornější představu slouží diagram tříd v příloze 11.

4.1.2 Třída MainWindow

Celá aplikace se ovládá z okna MainWindow, které vidíme na obrázku 6:

Obrázek 6: NCIP Initiator



Zdroj: vlastní zpracování

Na obrázku 6 je vidět hlavní a jediné okno klientské aplikace (MainWindow). ComboBox v pravém horním rohu nabízí uživateli výběr, pro kterou službu chce zprávu vytvořit. Zadá data potřebná pro zvolenou službu, popřípadě zaškrtne nepovinné elementy. Tlačítko **Vytvoř zprávu** vygeneruje zprávu pro horní TextBox ze zadaných dat. Tato zpráva se zde dá ještě ručně upravovat. Tlačítko **Odešli** odešle zprávu na adresu, kterou lze vyplnit vedle tlačítka. Spodní TextBox slouží pro obdržení odpovědi od druhé aplikace (Responder).

4.1.3 Metoda loadSettings

Metoda **loadSettings()** nastaví objekty viditelné v GUI pro zahájení práce s aplikací. Konkrétně se nastaví název okna na **NCIPInitiator**, TextBox získá scroll bar, **textBoxAddress** se nastaví v základě na localhost. **ComboBoxServices** dostane hodnoty

pro služby, které Initiator umožňuje odeslat. Nakonec se objekty pro jednotlivé služby skryjí, aby nemátly uživatele. Viz odrojový kód 1 **Metoda loadSettings()**.

Zdrojový kód 1: Metoda loadSettings()

```
private void loadSettings()
{
    this.Title = "NCIPInitiator";
    textBoxToSend.HorizontalScrollBarVisibility =
ScrollBarVisibility.Visible;
    textBoxToSend.VerticalScrollBarVisibility =
ScrollBarVisibility.Visible;
    textBoxToSend.TextWrapping = TextWrapping.NoWrap;
    textBoxAnswer.HorizontalScrollBarVisibility =
ScrollBarVisibility.Visible;
    textBoxAnswer.VerticalScrollBarVisibility =
ScrollBarVisibility.Visible;
    textBoxAnswer.TextWrapping = TextWrapping.NoWrap;
    textBoxAddress.Text =
"http://localhost:8080/web/NCIPResponder";

    comboBoxServices.Items.Add("Lookup User Service");
    comboBoxServices.Items.Add("Lookup Item Service");

    textBoxUserId.Visibility = Visibility.Hidden;
    textBoxItemId.Visibility = Visibility.Hidden;
    checkBoxLoanedItemsDesired.Visibility = Visibility.Hidden;
    checkBoxRequestedItemsDesired.Visibility =
Visibility.Hidden;
    checkBoxUserFiscalAccountDesired.Visibility =
Visibility.Hidden;
}
```

Zdroj: vlastní zpracování

4.1.4 Metoda comboBoxServices_SelectionChanged

Po zvolení služby v **comboBoxServices** se odkryjí objekty pro zadávání hodnot pro zprávu vybrané služby. V případě **Lookup User Service** to jsou objekty CheckBox s nepovinnými elementy a TextBox pro **UserId**. U **Lookup Item Service** se odkryje pouze TextBox pro **ItemId,** viz zdrojový kód 2.

Zdrojový kód 2: Metoda comboBoxServices_SelectionChanged()

```
private void comboBoxServices_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
```

```
    if (comboBoxServices.SelectedValue.Equals("Lookup User
Service"))
    {
        textBoxItemId.Visibility = Visibility.Hidden;

        textBoxUserId.Visibility = Visibility.Visible;
        checkBoxLoanedItemsDesired.Visibility =
Visibility.Visible;
        checkBoxRequestedItemsDesired.Visibility =
Visibility.Visible;
        checkBoxUserFiscalAccountDesired.Visibility =
Visibility.Visible;
    }
    if(comboBoxServices.SelectedValue.Equals("Lookup Item
Service"))
    {
        textBoxUserId.Visibility = Visibility.Hidden;
        checkBoxLoanedItemsDesired.Visibility =
Visibility.Hidden;
        checkBoxRequestedItemsDesired.Visibility =
Visibility.Hidden;
        checkBoxUserFiscalAccountDesired.Visibility =
Visibility.Hidden;

        textBoxItemId.Visibility = Visibility.Visible;
    }
}
```

Zdroj: vlastní zpracování

4.1.5 Metoda buttonGenerate_Click

Po kliknutí na tlačítko **buttonGenerate** se podle zvolené služby v **comboBoxServices** zavolá konkrétní metoda pro danou službu z objektu **toXml**, tedy ze třídy ToXml. Pokud je zvolena služba **Lookup User Service**, zavolá se **toXmlLookupUserService(),** která očekává čtyři vstupní parametry. Prvním je **UserId**, zbylé tři ukazují, zda uživatel zaškrtl jeden ze tří nabízených nepovinných elementů. Viz zdrojový kód 3.

Zdrojový kód 3: Metoda buttonGenerate_Click()

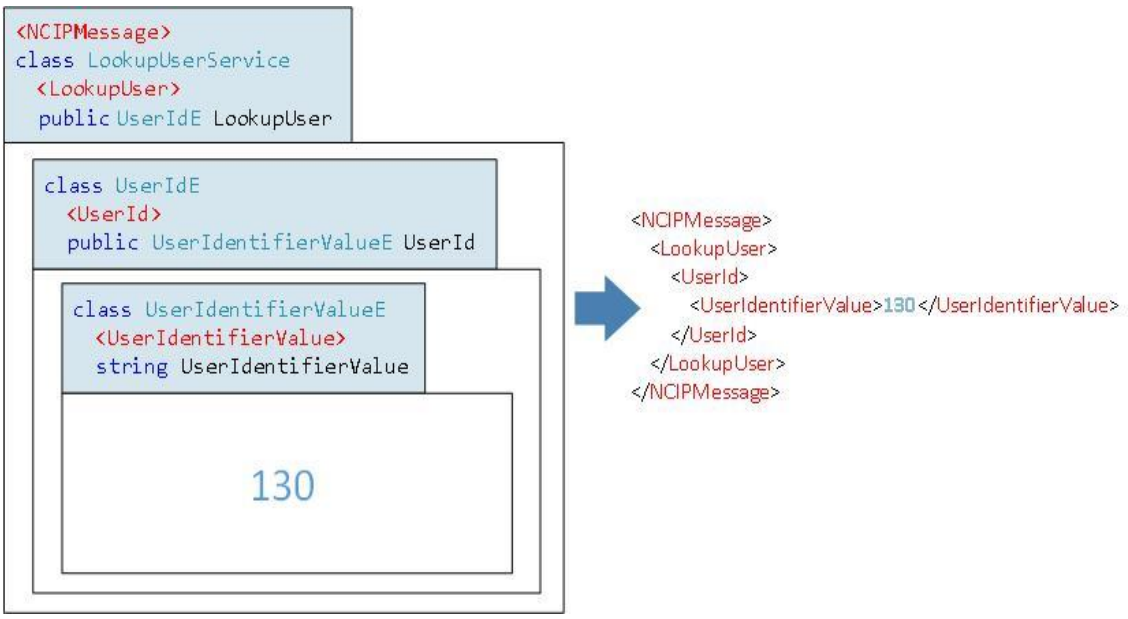
```
private void buttonGanerate_Click(object sender,
RoutedEventArgs e)
{
    if (comboBoxServices.SelectedValue.Equals("Lookup User
Service"))
    {
        textBoxToSend.Text =
toXml.toXmlLookupUserService(textBoxUserId.Text,

checkBoxLoanedItemsDesired.IsChecked.Value,
checkBoxRequestedItemsDesired.IsChecked.Value,
checkBoxUserFiscalAccountDesired.IsChecked.Value);
    }
    if (comboBoxServices.SelectedValue.Equals("Lookup Item
Service"))
    {
        textBoxToSend.Text =
toXml.toXmlLookupItemService(textBoxItemId.Text);
    }
}
```

Zdroj: vlastní zpracování

4.1.6 XmlSerializer

Pro využití XmlSerializer je třeba správně použít nadefinované objekty. Jejich posloupnost vidíme na obrázku 7:

Obrázek 7: Objekty pro XmlSerializer



Zdroj: vlastní zpracování

Na obrázku 7 vidíme, že hlavní třídou je LookupUserService, kterou je potřeba předat konstruktoru XmlSerializer. Podle ní bude XmlSerializer tvořit výsledné XML. Je důležité si uvědomit, že ElementName této třídy je hlavním elementem XML. V případě NCIP tedy **NCIPMessage**. Název její vlastnosti **LookupUser** je dalším elementem, který obsahuje element **NCIPMessage**. Vlastnost **LookupUser** pracuje s třídou UserIdE, která má zase vlastnost **UserId**. Tato vlastnost bude dalším elementem, který je pod elementem **LookupUser**. Posledním elementem je **UserIdentifierValue**, který už obsahuje konkrétní hodnotu např.: 130. V pravé části obrázku je vidět výsledné XML z tohoto seřazení objektů do sebe. Zdrojové kódy jsou znázorněny v části 4. 1. 7. Třída LookupUserService. Takovéto XML, však ještě není kompletní, takže by nebylo zpracováno. Viz obrázek 8.

Obrázek 8: Ukázka NCIP zprávy s povinnými náležitostmi.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:NCIPMessage
  xmlns:ns1="http://www.niso.org/2008/ncip"
  ns1:version="http://www.niso.org/schemas/ncip/v2_0/imp1/xsd/ncip_v2_0.xsd">
  <ns1:LookupUser>
    <ns1:UserId>
      <ns1:UserIdentifierValue>1</ns1:UserIdentifierValue>
    </ns1:UserId>
  </ns1:LookupUser>
</ns1:NCIPMessage>
```

Zdroj: vlastní zpracování

Na obrázku 8 je fialovým písmem zvýrazněna XML hlavička. Dále zde musí být prefixy elementů, tedy ns1. Element **NCIPMessage** musí obsahovat také attribute version (zvýrazněn zeleným písmem). Žlutým písmem je zvýrazněn Namespace. Pokud by zvýrazněné prvky chyběly, byla by komunikace neúspěšná a byl by navrácen Problem element.

4.1.7 Třída LookupUserService

Tato třída představuje zároveň hlavní element XML zprávy. To znamená, že ElementName je nastavený na **NCIPMessage** a Namespace na **http://www.niso.org/2008/ncip**. Zároveň je potřeba přidat attribute **version**. Ve zdrojovém kódu 4 vidíme vlastnost **LookupUser** – právě tento název je název druhého elementu po **NCIPMessage**.

Zdrojový kód 4: Třída LookUserService

```
[Serializable()]
[XmlRoot(ElementName = "NCIPMessage", Namespace =
"http://www.niso.org/2008/ncip")]
public class LookupUserService
{
     [XmlAttribute("version", Namespace =
"http://www.niso.org/2008/ncip", Form =
XmlSchemaForm.Qualified)]
    public string version { get; set; }

    private UserIdE userId;
    public UserIdE LookupUser
    {
        get { return userId; }
        set { userId = value; }
```

```
    }
}
```

Zdroj: vlastní zpracování

V kódu 4 vidíme, že potřebujeme objekt UserIdE. Ten je definován níže ve zdrojovém kódu 5:

Zdrojový kód 5: Třída UserIdE

```
public class UserIdE
{
    private UserIdentifierValueE userIdentifierValue;
    public UserIdentifierValueE UserId
    {
        get { return userIdentifierValue; }
        set { userIdentifierValue = value; }
    }
}
```

Zdroj: vlastní zpracování

Název vlastnosti **UserId** je třetím elementem v XML zprávě.

Níže je kód třídy UserIdentifierValueE, která je potřeba jako poslední element. Název elementu bude **UserIdentifierValue**.

Zdrojový kód 6: Třída UserIdentifierValue

```
public class UserIdentifierValueE
{
    private string userId;
    public string UserIdentifierValue
    {
        get { return userId; }
        set { userId = value; }
    }
}
```

Zdroj: vlastní zpracování

4.1.8 Třída ToXml

Třída ToXml obstarává vytvoření XML zpráv ve správném formátu. V konstruktoru se nastaví prefix a Namespace budoucí XML zprávy (zdrojový kód 7). Zde je důležité si uvědomit, že odkaz v Namespace musí být totožný s odkazem v XMLRoot (např.: [XmlRoot(ElementName = "NCIPMessage", Namespace = "http://www.niso.org/2008/ncip")]).

Zdrojový kód 7: Konstruktor ToXml()

```
public ToXml()
{
    namespaces.Add("ns1", "http://www.niso.org/2008/ncip");
    lookupUserService.version =
"http://www.niso.org/schemas/ncip/v2_0/imp1/xsd/ncip_v2_0.xsd"
;
    lookupItemService.version =
"http://www.niso.org/schemas/ncip/v2_0/imp1/xsd/ncip_v2_0.xsd"
;
}
```

Zdroj: vlastní zpracování

Třída ToXml obsahuje metody pro dvě implementované služby v tomto Initiatoru. Níže vidíme příklad jedné z metod (zdrojový kód 8). Tato metoda konkrétně vytváří zprávu pro službu **Lookup User Service**. Lze zde vidět použití objektů, potřebných pro vytvoření správného XML pomocí XmlSerializer.

Zdrojový kód 8: Metoda toXmlLookupUserService()

```
public string toXmlLookupUserService(string
userIdentifierValuer, bool LoanedItemsDesired,
            bool RequestedItemsDesired, bool
UserFiscalAccountDesired)
{
    XmlSerializer SerializerObj = new
XmlSerializer(typeof(LookupUserService));
    string ret = "";

    UserIdentifierValueE userIdentifierValuerE = new
UserIdentifierValueE();
    userIdentifierValuerE.UserIdentifierValue =
userIdentifierValuer;

    UserIdE userId = new UserIdE();
    userId.UserId = userIdentifierValuerE;

    lookupUserService.LookupUser = userId;

    var sw = new UTF8StringWriter();
    SerializerObj.Serialize(sw, lookupUserService,
namespaces);
    ret = sw.ToString();

    if (UserFiscalAccountDesired)
```

```
    { ret =
AddLookUserElements.addUserFiscalAccountDesired(ret); }
    if (RequestedItemsDesired)
    { ret = AddLookUserElements.addRequestedItemsDesired(ret);
}
    if(LoanedItemsDesired)
    { ret = AddLookUserElements.addLoanedItemsDesired(ret); }

    return ret;
}
```

Zdroj: vlastní zpracování

Soubor ToXml.cs, který obsahuje třídu ToXml, obsahuje i třídu UTF8StringWriter, která řeší problém třídy XmlSerializer využité ve třídě ToXml, protože tvořila zprávy v utf-16. NCIP totiž vyžaduje utf-8, takže toto bylo vyřešeno pomocí třídy UTF8StringWriter, která je znázorněna ve zdrojovém kódu 9:

Zdrojový kód 9: Třída UTF8StringWriter

```
public sealed class UTF8StringWriter : StringWriter
{
    public override Encoding Encoding { get { return
Encoding.UTF8; } }
}
```

Zdroj: vlastní zpracování

4.1.9 Metoda sendMessage

Po kliknutí na tlačítko **Odešli** se přes metodu **buttonSend_Click()** zavolá metoda **sendMessage()**, která je zobrazena ve zdrojovém kódu 10.

Zdrojový kód 10: Metoda sendMessage()

```
private void sendMessage()
{
    string xmlMessage = textBoxToSend.Text;
    string destinationUrl = textBoxAddress.Text;
    HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(destinationUrl);
    byte[] bytes;
    bytes = System.Text.Encoding.ASCII.GetBytes(xmlMessage);
    request.ContentType = "text/xml; encoding='utf-8'";
    request.ContentLength = bytes.Length;
    request.Method = "POST";
    Stream requestStream = request.GetRequestStream();
    requestStream.Write(bytes, 0, bytes.Length);
```

```
    requestStream.Close();
    HttpWebResponse response;
    response = (HttpWebResponse)request.GetResponse();
    if (response.StatusCode == HttpStatusCode.OK)
    {
        Stream responseStream = response.GetResponseStream();
        string answer = new
StreamReader(responseStream).ReadToEnd();
        showAnswer(answer);
    }
}
```

Zdroj: vlastní zpracování

Metoda převezme zprávu z **textBoxToSend** a odešle ji na adresu, která byla vyplněna v **textBoxAddress**. Na závěr metody se pak zavolá metoda **showAnswer()**, která zpracuje odpověď od Responderu.

Zdrojový kód 11: Metoda showAnser()

```
private void showAnswer(string answer)
{
    textBoxAnswer.Text = formatXml(answer);
}
```

Zdroj: vlastní zpracování

Ke zpracování se použije metoda **formatXml()**, která se pokusí utvořit z příchozího řetězce přehledný a odřádkovaný řetězec, který se zobrazí v **textBoxAnswer** (zdrojový kód 12).

Zdrojový kód 12: Metoda formatXml()

```
private string formatToXml(string xml)
{
    try
    {
        XDocument doc = XDocument.Parse(xml);
        return doc.ToString();
    }
    catch (Exception)
    {
        return xml;
    }
}
```

Zdroj: vlastní zpracování

4.2 Responder

Initiator by nebylo možné otestovat bez odpovídající aplikace Responder. Už není výhodné, aby programátor dělal Responder vlastními silami. XC NCIP Toolkit ušetří programátorovi hodně práce. Po stažení XC NCIP Toolkit se lze inspirovat už v hotových kódech. Jsou zde např.: zdrojové kódy konektorů k různým knihovním systémům jako jsou např.: Aleph, Evergreen, Koha a jiné. Za větší pozornost však stojí vzorová aplikace NCIP V2 Dummy Responder, která se dá pro ukázku NCIP protokolu použít. Tato webová aplikace slouží k testování zpráv posílaných protokolem NCIP. Lze ji stáhnout z eXtensible Catalog Organization (2016).

Pro využití aplikace NCIP V2 Dummy Responder bylo zapotřebí přidání objektů, které lze vidět v diagramu tříd v příloze 12. Jsou to tyto třídy:

1. Item – slouží pro uchování dat o Item
2. User – slouží pro uchování dat o User
3. Fee – slouží pro uchování dat o Fee
4. DPDatabase – slouží pro naplnění výše uvedených objektů daty z databáze

Takto připravené objekty jsou využité v upravených třídách DummyLookupItemService a DummyLookupUserService.

Při buildování projektu ve vývojářském prostředí NetBeans se naskytly drobné komplikace. Ze souboru pom.xml bylo potřeba odstranit:

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.1-beta-1</version>
</plugin>
```

Docházelo zde ke konfliktům verze Mavenu a JDK.

4.2.1 Třída DPDatabase

Pro získání dat z databáze by stačilo, uchovat je přímo v proměnných v kódu, ale tato metoda by nebyla přehledná ani praktická. Proto vznikla třída DPDatabase, která obsahuje metody pro získávání dat z databáze. Jako databáze bylo použito MSSQL.

Cílem práce není tvorba knihovního systému, ale předvedení možné implementace protokolu NCIP. Vytvářet složitou databázi není potřeba, viz příloha 10.

V opravdovém knihovním systému by samozřejmě mnohem více tabulek, např.: by zde byla navíc tabulka Work, která by v sobě uchovávala data o dílech. Každé dílo může mít neomezeně svazků, ale pro zjednodušení se zde uchovávají pouze svazky v tabulce Item. Chybí zde i různá metadata, která pro ukázku nebyla potřebná např.: datum registrace uživatele. V tabulce User jsou uchovány pouze základní údaje o uživateli. V Agency jsou základní údaje o knihovně potřebné pro ukázku NCIP protokolu. Tabulka Loan má v sobě uložena data o výpůjčkách uživatelů. Poslední tabulka Fee nese údaje o poplatcích uživatele. V tabulce Loan jsou vedeny výpůjčky. Tabulka Request slouží pro zaznamenání rezervací. V tabulce Account je zaznamenán stav účtu uživatele.

Jedna z metod třídy DPDatabase je např.: **getUser()**, která po přijetí id uživatele, vrátí objekt User, viz příloha 5, který naplní údaji z databáze, viz zdrojový kód 13.

Zdrojový kód 13: Metoda getUserName()

```
public static User getUser(String userID) throws SQLException,
ClassNotFoundException
    {
        User result = new User();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "SELECT * FROM [DP].[dbo].[User] WHERE [id] = " +
userID;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
           result.setUserId(rs.getString("id"));
           result.setUserFirstName(rs.getString("firstname"));
           result.setUserLastName(rs.getString("lastname"));
           result.setUserAddressCity(rs.getString("address_city"));

result.setUserAddressStreetNumber(rs.getString("address_street_number"));
           result.setUserAddressZipCode(rs.getString("address_zip_code"));
           result.setUserDateOfBirth(rs.getString("date_of_birth"));
        }
        st.close();
        return result;
    }
```

Zdroj: vlastní zpracování

Celou třídu DPDatabase se všemi metodami lze nalézt v příloze 6.

4.2.2 Třída DummyLookupUserService

Třídu DummyLookupUserService použije Responder v případě, že Initiator pošle dotaz na službu **Lookup User**. Struktura a popis zprávy pro službu **Lookup User** je popsán v příloze 2 v tabulce **Lookup User**. Zde dochází ke shromáždění a uprávě dat pro objekt **LookupUserResponseData**, ze kterého se vytvoří XML pro odpověď. Připravit data pro odpověď by nebylo možné bez předchozích dat. Responder by nevěděl např.: o kterém uživateli data připravit. Z tohoto důvodu je jedním ze vstupních parametrů **LookupUserInitiationData**, který obsahuje data z příchozí zprávy. Z tohoto objektu si pak lze např.: pomocí **getUserId().getUserIdentifierValue()** vytáhnout id o příchozím **User**. Třída DummyLookupUserService je v příloze 7. Pro každý nepovinný element je zde jedna podmínka if, která zpracuje konkrétní data. Jeden CheckBox, který lze v klientu zaškrtnout, odpovídá tedy jedné podmínce.

4.2.3 Třída DummyLookupItemService

Třídu DummyLookupItemService použije Responder v případě, že Initiator pošle dotaz na službu **Lookup Item**. Struktura a popis zprávy pro službu **Lookup Item** jsou popsány v příloze 2 v tabulce **Lookup Item**. Dochází zde ke shromažďování a upraveně dat pro objekt **LookupItemResponseData**, ze kterého se vytvoří XML pro odpověď. Připravit data pro odpověď by nebylo možné bez předchozích dat. Responder by nevěděl např.: o kterém svazku data připravit. Z tohoto důvodu je jedním ze vstupních parametrů **LookupItemInitiationData**, který obsahuje data z příchozí zprávy. Z tohoto objektu si pak lze např.: pomocí **getItemId().getItemIdentifierValue()** vytáhnout id o příchozím Item. Třída DummyLookupItemService je v příloze 8.

4.3 Test komunikace aplikací

4.3.1 Test Lookup User

Test pro službu **Lookup User byl** proveden odesláním zprávy klientem, viz obrázek 9.

Obrázek 9: Testovací zpráva Lookup User

```
<?xml version="1.0" encoding="utf-8"?>
<ns1:NCIPMessage ns1:version="http://www.niso.org/schemas/ncip/v2_0/imp1/xsd/
ncip_v2_0.xsd" xmlns:ns1="http://www.niso.org/2008/ncip">
 <ns1:LookupUser>
  <ns1:UserId>
   <ns1:UserIdentifierValue>2</ns1:UserIdentifierValue>
  </ns1:UserId>
  <ns1:LoanedItemsDesired/>
  <ns1:RequestedItemsDesired/>
  <ns1:UserFiscalAccountDesired/>
 </ns1:LookupUser>
</ns1:NCIPMessage>
```

Zdroj: vlastní zpracování

První testovací zprávou se Initiator dotázal na informace o uživateli s id 2. Zpráva zároveň obsahovala tři nepovinné elementy **<ns:1LoanedItemsDesired/>**, **<ns1:RequestedItemsDesired/>** a **<ns1:UserFiscalAccountDesired/>**.

Celá odpověď na zprávu z obrázku 9 lze vidět v příloze 9. Jednotlivé části jsou zde barevně odlišeny:

1. <ns1:UserFiscalAccountDesired/> – zelená
2. <ns:1LoanedItemsDesired/> – fialová
3. <ns1:RequestedItemsDesired/> – hnědá

V první části odpovědi je uvedeno id **User**, které musí být stejné jako odchozí, tedy 2 viz obrázek 10.

Obrázek 10: Příchozí id User

```
<ns1:UserId>
   <ns1:UserIdentifierValue>2</ns1:UserIdentifierValue>
  </ns1:UserId>
```

Zdroj: vlastní zpracování

Následují údaje požadované nepovinným elementem **<ns1:UserFiscalAccountDesired/>**, kde vidíme stav účtu **User** a poplatky. Na obrázku 11 jsou pro ukázku zobrazeny informace o jednom poplatku. Vidíme zde především tyto informace: název svazku, autor, výše poplatku, měna a jeho popis.

Obrázek 11: Příchozí informace o poplatku

```
<ns1:FiscalTransactionType ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscaltransactiontype/
fiscaltransactiontype.scm">Fine</ns1:FiscalTransactionType>
      <ns1:ValidFromDate>2016-01-23T11:52:14.723Z</ns1:ValidFromDate>
      <ns1:ValidToDate>2017-03-08T11:52:14.723Z</ns1:ValidToDate>
      <ns1:Amount>
       <ns1:CurrencyCode ns1:Scheme="http://www.bsi-global.com/Technical+Information/Publications/
_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
       <ns1:MonetaryValue>50</ns1:MonetaryValue>
      </ns1:Amount>
      <ns1:FiscalTransactionDescription>poplatek za pozdní vrácení</ns1:FiscalTransactionDescription>
      <ns1:ItemDetails>
       <ns1:ItemId>
        <ns1:ItemIdentifierValue>3</ns1:ItemIdentifierValue>
       </ns1:ItemId>
       <ns1:BibliographicDescription>
        <ns1:Author>Karel Čapek</ns1:Author>
        <ns1:Title>Krakatit</ns1:Title>
       </ns1:BibliographicDescription>
       <ns1:DateCheckedOut>2015-12-24T11:52:14.723Z</ns1:DateCheckedOut>
       <ns1:DateDue>2016-01-18T11:52:14.723Z</ns1:DateDue>
       <ns1:DateReturned>2016-01-23T11:52:14.723Z</ns1:DateReturned>
      </ns1:ItemDetails>
    </ns1:FiscalTransactionInformation>
```

Zdroj: vlastní zpracování

Na obrázku 12 je zobrazena jedna výpůjčka z odpovědi. Vidíme zde například název titulu nebo typ výpůjčky.

Obrázek 12: Příchozí informace o výpůjčce

```
<ns1:LoanedItemsCount>
   <ns1:CirculationStatus ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/circulationstatus/
circulationstatus.scm">On Loan</ns1:CirculationStatus>
   <ns1:LoanedItemCountValue>2</ns1:LoanedItemCountValue>
  </ns1:LoanedItemsCount>
  <ns1:LoanedItem>
   <ns1:ItemId>
    <ns1:ItemIdentifierValue>1</ns1:ItemIdentifierValue>
   </ns1:ItemId>
   <ns1:DateDue>2016-06-01T11:32:41.179Z</ns1:DateDue>
   <ns1:Amount>
    <ns1:CurrencyCode ns1:Scheme="http://www.bsi-global.com/Technical+Information/Publications/
_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
    <ns1:MonetaryValue>0</ns1:MonetaryValue>
   </ns1:Amount>
   <ns1:Title>Máj</ns1:Title>
   <ns1:Ext>
    <ns1:RenewalCount>0</ns1:RenewalCount>
    <ns1:DateCheckedOut>2016-03-08T12:32:41.201Z</ns1:DateCheckedOut>
   </ns1:Ext>
  </ns1:LoanedItem>
```

Zdroj: vlastní zpracování

Na obrázku 13 jsou zobrazeny informace rezervací jako např.: id rezervace a **Item**, datum vytvoření, datum, kdy je rezervace připravena k vyzvednutí, datum, do kdy platí rezervace, název titulu a místo k vyzvednutí.

Obrázek 13: Příchozí informace o rezervaci

```
<ns1:RequestedItem>
   <ns1:RequestId>
    <ns1:RequestIdentifierValue>1</ns1:RequestIdentifierValue>
   </ns1:RequestId>
   <ns1:ItemId>
    <ns1:ItemIdentifierValue>1</ns1:ItemIdentifierValue>
   </ns1:ItemId>
   <ns1:RequestType ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/requesttype/
requesttype.scm">Loan</ns1:RequestType>
   <ns1:RequestStatusType ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/requeststatustype/
requeststatustype.scm">Available For Pickup</ns1:RequestStatusType>
   <ns1:DatePlaced>2016-02-01T12:40:17.500Z</ns1:DatePlaced>
   <ns1:PickupDate>2016-04-01T11:40:17.500Z</ns1:PickupDate>
   <ns1:PickupLocation>Knihovna ČB</ns1:PickupLocation>
   <ns1:PickupExpiryDate>2016-06-01T11:40:17.500Z</ns1:PickupExpiryDate>
   <ns1:Title>Babička</ns1:Title>
  </ns1:RequestedItem>
```

Zdroj: vlastní zpracování

Na konci odpovědi jsou zobrazeny ještě dodatečné informace o **User**, viz obrázek 14.

Obrázek 14: Příchozí informace o User

```
<ns1:UserOptionalFields>
   <ns1:NameInformation>
    <ns1:PersonalNameInformation>
     <ns1:UnstructuredPersonalUserName>Jakub Devátý</ns1:UnstructuredPersonalUserName>
    </ns1:PersonalNameInformation>
   </ns1:NameInformation>
   <ns1:DateOfBirth>1991-01-01T12:40:17.357Z</ns1:DateOfBirth>
  </ns1:UserOptionalFields>
```

Zdroj: vlastní zpracování

4.3.2 Test Lookup Item

Druhý test byl proveden pro službu **Lookup Item**. Zpráva měla strukturu, která je zobrazena na obrázku 15.

Obrázek 15: Testovací zpráva Lookup Item

```
<?xml version="1.0" encoding="utf-8"?>
<ns1:NCIPMessage ns1:version="http://www.niso.org/schemas/ncip/v2_0/imp1/xsd/ncip_v2_0.xsd"
xmlns:ns1="http://www.niso.org/2008/ncip">
 <ns1:LookupItem>
  <ns1:ItemId>
   <ns1:ItemIdentifierValue>2</ns1:ItemIdentifierValue>
  </ns1:ItemId>
 </ns1:LookupItem>
</ns1:NCIPMessage>
```

Zdroj: vlastní zpracování

Z obrázku je vidět, že se Initiator dotazuje na **Item** s id 2.

Obrázek 16: Odpověď na testovací zprávu Lookup Item

```
<ns1:NCIPMessage xmlns:ns1="http://www.niso.org/2008/ncip" xmlns:ns2="http://oclc.org/WCL/ncip/2011/
extensions" xmlns:ns3="http://www.oclc.org/ncip/usernote/2012" ns1:version="http://www.niso.org/schemas/
ncip/v2_0/imp1/xsd/ncip_v2_0.xsd">
 <ns1:LookupItemResponse>
  <ns1:ItemId>
   <ns1:ItemIdentifierValue>2</ns1:ItemIdentifierValue>
  </ns1:ItemId>
  <ns1:ItemOptionalFields>
   <ns1:BibliographicDescription>
    <ns1:Author>Karel Hynek Mácha</ns1:Author>
    <ns1:BibliographicRecordId>
     <ns1:BibliographicRecordIdentifier>1531</ns1:BibliographicRecordIdentifier>
     <ns1:AgencyId>Knihovna ČB</ns1:AgencyId>
    </ns1:BibliographicRecordId>
    <ns1:Edition>2.</ns1:Edition>
    <ns1:PublicationDate>2015-11-11</ns1:PublicationDate>
    <ns1:Publisher>Vydavatel v Praze</ns1:Publisher>
    <ns1:Title>Máj</ns1:Title>
    <ns1:Language ns1:Scheme="http://lcweb.loc.gov/standards/iso639-2/bibcodes.html">cze</ns1:Language>
   </ns1:BibliographicDescription>
   <ns1:CirculationStatus ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/circulationstatus/
circulationstatus.scm">Available On Shelf</ns1:CirculationStatus>
   <ns1:ItemDescription>
    <ns1:CallNumber>10</ns1:CallNumber>
    <ns1:HoldingsInformation>
     <ns1:UnstructuredHoldingsData>holder</ns1:UnstructuredHoldingsData>
    </ns1:HoldingsInformation>
    <ns1:NumberOfPieces>1</ns1:NumberOfPieces>
   </ns1:ItemDescription>
  </ns1:ItemOptionalFields>
 </ns1:LookupItemResponse>
</ns1:NCIPMessage>
```

Zdroj: vlastní zpracování

Na obrázku 16 jsou vidět příchozí informace o dotazované knize: název, autor, datum vydání nebo její aktuální stav v knihovně.

ZÁVĚR

Cílem práce bylo popsat protokol NISO Circulation Interchange Protocol (NCIP), který slouží pro komunikaci mezi knihovními systémy a navrhnout vlastní řešení implementace. Byl vytvořen Initiator v jazyce C# pomocí Visual Studio 2015. Pro práci s formátem XML byl zvolen XmlSerializer, který nabízí objektové zpracování a vytvoření XML dokumentů. Pro test byla potřeba i druhé aplikace – Responder, aby bylo možné provést test komunikace přes NCIP. Responder byl vytvořen z XC NCIP Toolkit verze 2, konkrétně z NCIP V2 Dummy Responder. Tato aplikace licence MIT/X11 byla upravena tak, aby byla pro test komunikace schopná komunikovat s databází MSSQL, která byla napojena na Microsoft SQL Server 2014.

Testování bylo provedeno na službách Lookup User a Lookup Item. Při každém testu byly zjištěny drobné nedostatky, které byly následně opraveny. Například při načítání dat z databáze se data načítala do objektu, který byl následně uložen do listu. Tento objekt byl vytvořen před cyklem a tím docházelo k tomu, že všechny objekty v listu měly stejné hodnoty, jako měl mít poslední objekt. Po opravení těchto nedostatků Initiator úspěšně odeslal zahajující zprávu pro Responder, který zpracoval a připravil data z databáze pro odpověď. Odpověď byla úspěšně zobrazena v aplikaci Initiator, kde všechna data souhlasila s databází pro Responder.

Všechny předem stanovené cíle byly splněny.

Protokol NCIP je využíván v mnoha knihovních systémech, např.: Tritius, Aleph nebo Koha. Práci lze využít jako návod pro implementaci protokolu NCIP do dalších systémů.

Tuto práci by bylo možné rozvíjet implementací více služeb, jak na straně klienta tak serveru. Při využití Responder v praxi by implementace pokračovala napojením na větší a plnohodnotnější databázi, jakou používá knihovní systém. Bylo by nevhodné, kdyby byly termíny překládány doslovně do češtiny, protože jsou tak popisovány v dokumentaci k protokolu, a proto se autor snažil používat pojmy v původním, tedy anglickém jazyce.

Summary and keywords

This work tries to describe the possibility of implementation of Protocol NISO Circulation Interchange Protocol. In the theoretical part will be mentioned the history, structure and importance of the protocol. There will also be mentioned the mark-up language XML and protocol HTTP. In the practical part will be created application in C # which communicates through NCIP protocol. The XML processing ensures XmlSerializer class. For testing will be modified NCIP V2 Dummy Responder, which uses Java XC NCIP Toolkit.

Keywords: C#, Java, NCIP, XmlSerializer, XML

Seznam použitých zdrojů

1. BRADLEY, N. (2000). *XML kompletní průvodce.* Praha: Grada Publishing s. r. o. ISBN: 80-7169-949-7
2. eXtensible Catalog Organization. (2016). *NCIP Toolkit.* Dostupné z WWW: http://www.extensiblecatalog.org/
3. KOSEK, J. (2000a). *XML*. Dostupné z WWW: http://www.kosek.cz/clanky/swn-xml/uvod.html
4. KOSEK, J. (2000b). *XML pro každého podrobný průvodce.* Praha: Grada Publishing s. r. o. ISBN: 80-7168-860-1
5. KOSEK, J. (2000c). *Základy protokolu HTTP.* Dostupné z WWW: http://www.kosek.cz/clanky/iweb/05.html
6. NCIP STANDING COMMITEE. (2011). *Introduction to NCIP*. Dostupné z WWW: http://www.ncip.info/introduction-to-ncip.html
7. NISO. (2015a). *Background and history*. Dostupné z WWW: http://www.niso.org/workrooms/ncip
8. NISO. (2015b). *Historical note*. Dostupné z WWW: http://digital.lib.umd.edu/archivesum/actions.DisplayEADDoc.do?source=MdU.ead.histms.0113.xml&style=ead
9. NISO. (2012c). *NISO Circulation Interchange Part1: Protocol (NCIP).* Dostupné z WWW: http://www.niso.org/apps/group_public/download.php/8965/z39-83-2-2012_NCIP.pdf
10. NISO. (2012d). *NISO Circulation Interchange Protocol (NCIP) Part2: Implementation Profile 1*. Dostupné z WWW: http://www.niso.org/apps/group_public/download.php/8966/z39-83-1-2012_NCIP.pdf
11. NOVÁK, D. (2006). *Metody udržování stavových informací v protokolu http.* Praha. (Bakalářská práce). Dostupné z WWW: https://www.vse.cz/vskp/721_metody_udrzovani_stavovych_informaci_v_protokolu_http
12. YOUNG, J., M. (2002). *XML krok za krokem.* Praha: Mobil Media a.s. ISBN: 80-86593-28-2

Seznam obrázků

Seznam zdrojových kódů

Seznam tabulek

Seznam příloh

Přílohy

Příloha 1: Shrnutí typů služeb a odpovědí (NISO, 2015d)

	Service	
Service type	Initiation message	Response message
LOOKUP (vyhledávání)	Lookup agency	Lookup agency Response
	Lookup Item	Lookup Item Response
	Lookup Request	Lookup Request Response
	Lookup User	Lookup User Response
	Lookup Version	Lookup Version Response
	Lookup Item Set	Lookup Item Set Response
Update (změna)	Accept Item	Accept Item Response
	Check In Item	Check In Item Response
	Check Out Item	Check Out Item Response
	Undo Check Out Item	Undo Check Out Item Response
	Create Agency	Create Agency Response
	Create Item	Create Item Response
	Create User	Create User Response
	Create User Fiscal Transaction	Create User Fiscal Transaction Response
	Delete Item	Delete Item Response
	Delete User	Delete User Response
	Recall Item	Recall Item Response
	Cancel Recall Item	Cancel Recall Item Response
	Renew Item	Renew Item Response
	Report Circulation Status Change	Report Circulation Status Change Response
	Request Item	Request Item Response
	Cancel Request Item	Cancel Request Item Response
	Send User Notice	Send User Notice Response

	Update Agency	Update Agency Response
	Update Circulation Status	Update Circulation Status Response
	Update Item	Update Item Response
	Update Request Item	Update Request Item Response
	Update User	Update User Response
Notification (oznámení)	Agency Created	Agency Created Response
	Agency Updated	Agency Updated Response
	Circulation Status Change Reported	Circulation Status Change Reported Response
	Circulation Status Updated	Circulation Status Updated Response
	Item Checked In	Item Checked In Response
	Item Checked Out	Item Checked Out Response
	Item Created	Item Created Response
	Item Recall Cancelled	Item Recall Cancelled Response
	Item Recalled	Item Recalled Response
	Item Received	Item Received Response
	Item Renewed	Item Renewed Response
	Item Request Cancelled	Item Request Cancelled Response
	Item Request Updated	Item Request Updated Response
	Item Requested	Item Requested Response
	Item Shipped	Item Shipped Response
	Item Updated	Item Updated Response
	User Created	User Created Response
	User Fiscal Transaction Created	User Fiscal Transaction Created Response
	User Notice Sent	User Notice Sent Response
	User Updated	User Updated Response

Zdroj: vlastní zpracování

Příloha 2: Lookup Service (NISO, 2015d)

Lookup User	
Použití	Služba požaduje konkrétní data o User, které odpovídající aplikace zná. Zahajující aplikace pošle Id User a elementy, pro které požaduje informace.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	
Lookup User Message	
Požadovaná data	{User Id or Authentication Input (R)}
Volitelná data	Initiation Header User Element Type (R) Loaned Items Desired Requested Items Desired User Fiscal Account Desired Ext
Lookup User Response	
Požadovaná data	User Id
Volitelná data	Response Header Ext
Pokud je vyžadováno	User Fiscal Account (R) Loaned Items Count (R) Loaned Item (R) Requested Items Count (R) Requested Item (R) User Optional Fields

Zdroj: vlastní zpracování

Lookup Agency	
Použití	Služba požaduje konkrétní data o Agency, která odpovídající aplikace zná. Zahajující aplikace pošle Id Agency a elementy, pro které požaduje informace.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	

Lookup Agency Initiation Message	
Požadovaná data	Agency Element Type (R) Agency Id
Volitelná data	Initiation Header Ext
Lookup Agency Response	
Požadovaná data	Agency Id
Volitelná data	Response Header Ext
Pokud je vyžadováno	Agency Address Information (R) Agency User Privilege Type (R) Application Profile Supported Type (R) Authentication Prompt (R) Consortium Agreement (R) Organization Name Information (R)

Zdroj: vlastní zpracování

Lookup Item	
Použití	Služba požaduje konkrétní data o Item, která odpovídající aplikace zná. Zahajující aplikace pošle Id Item a elementy, pro které požaduje informace.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	
Lookup Item Initiation Message	
Požadovaná data	{Item Id or Request Id}
Volitelná data	Initiation Header Item Element Type (R) Current Borrower Desired Current Requesters Desired Ext
Lookup Item Response	
Požadovaná data	{Item Id and/or

	Request Id}
Volitelná data	Response Header Ext
Pokud je vyžadováno	Hold Pickup Date Date Recalled Item Transaction Item Optional Fields

Zdroj: vlastní zpracování

Lookup Request	
Použití	Služba požaduje konkrétní data o Request, která odpovídající aplikace zná. Zahajující aplikace pošle Id Request a elementy, pro které požaduje informace.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	
Lookup Request Initiation Message	
Požadovaná data	{{Authentication Input (R) or User Id} Item Id Request Type}, or Request Id
Volitelná data	Initiation Header Request Element Type (R) Item Element Type (R) User Element Type (R) Ext
Lookup Request Response	
Požadovaná data	{Item Id, and/or Request Id}
Volitelná data	Response Header User Id Request Type Request Scope Type Request Status Type

	Hold Queue Position Shipping Information Earliest Date Needed Need Before Date

Zdroj: vlastní zpracování

Lookup User	
Použití	Služba požaduje konkrétní data o User, která odpovídající aplikace zná. Zahajující aplikace pošle Id User a elementy, pro které požaduje informace.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	
Lookup User Initiation Message	
Požadovaná data	{User Id or Authentication Input (R)}
Volitelná data	Initiation Header User Element Type (R) Loaned Items Desired Requested Items Desired User Fiscal Account Desired Ext
Lookup User Response	
Požadovaná data	User Id
Volitelná data	Response Header Ext
Pokud je vyžadováno	User Fiscal Account (R) Loaned Items Count (R) Loaned Item (R) Requested Items Count (R) Requested Item (R) User Optional Fields

Zdroj: vlastní zpracování

Lookup Version	
Použití	Služba požaduje informaci o verzích NCIP, které odpovídající aplikace podporuje.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	
Lookup Version Initiation Message	
Požadovaná data	From Agency Id To Agency Id
Volitelná data	None
Lookup Agency Response	
Požadovaná data	From Agency Id To Agency Id Version Supported (R)
Volitelná data	None

Zdroj: vlastní zpracování

Lookup Item Set	
Použití	Služba požaduje řadu dat o Items, která odpovídající aplikace zná. Zahajující aplikace pošle řadu identifikátorů Items a elementy, pro které požaduje informace.
Úspěšný výsledek	Aplikace vrátí požadovaná data.
Zprávy a datové elementy	
Lookup Item Set Initiation Message	
Požadovaná data	{Bibliographic Id, or Holdings Set Id, or Item Id}
Volitelná data	Item Element Type Current Borrower Desired Current Requesters Desired Maximum Items Count Next Item Token
Lookup Item Set Response	
Požadovaná data	Problem, or

	Bib Information
Volitelná data	Next Item Token
Pokud je vyžadováno	Current Borrower (inside Item Information) Current Requester (inside Bib Information)

Zdroj: vlastní zpracování

Příloha 3: Update Service (NISO, 2015d)

Renew Item	
Použití	Služba požaduje, aby odpovídající aplikace prodloužila Item pro User. Zahajující aplikace může navrhnout revizi termínu a potvrzení o výši poplatku. Zahajující aplikace také může požadovat data o Item nebo User, kterých se relace týká.
Úspěšný výsledek	Odpovídající aplikace prodlouží Item pro User a poskytne upravené datum pro vypůjčení. Dále pošle data pro požadované elementy.
Zprávy a datové elementy	
Renew Item Message	
Požadovaná data	{User Id or Authentication Input (R)} Item Id
Volitelná data	Initiation Header Mandated Action Item Element Type (R) User Element Type (R) Desired Date Due Desired Date For Return Acknowledged Fee Amount Paid Fee Amount Acknowledged Item Use Restriction Type (R) Ext
Renew Item Response	
Požadovaná data	Pending or Item Id

Volitelná data	Response Header User Id Date Due Date For Return Renewal Count Fiscal Transaction Information Ext
Pokud je vyžadováno	Item Optional Fields User Optional Fields
Odpověď pokud je problém	Response Header (optional) Problem (R, required) and Required Fee Amount (optional) Required Item Use Restriction Type (R, optional) Ext (optional)

Zdroj: vlastní zpracování

Request Item	
Použití	Služba požaduje, aby odpovídající aplikace umístila požadavek, zda je či není Item pro User okamžitě k dispozici. Zahajující aplikace indikuje typ požadavku. Zahajující aplikace může poskytnout informaci o poplatku za službu. Zahajující aplikace také může požadovat data o Item nebo User, kterých se relace týká.
Úspěšný výsledek	Odpovídající aplikace umístí požadavek a poskytne data, kde bude Item možné vyzvednout a datum vyzvednutí. Dále pošle data pro požadované elementy.
Zprávy a datové elementy	
Request Item Initiation Message	
Požadovaná data	{User Id or Authentication Input (R)} {Item Id (R) and/or Bibliographic Id (R)} Request Type Request Scope Type

Volitelná data	Initiation Header Mandated Action Request Id Item Optional Fields Shipping Information Earliest Date Needed Need Before Date Pickup Location Pickup Expiry Date Acknowledged Fee Amount Paid Fee Amount Acknowledged Item Use Restriction Type (R) Item Element Type (R) User Element Type (R) Ext
Request Item Response	
Požadovaná data	{Item Id and/or Request Id} User Id Request Type Request Scope Type}
Volitelná data	Response Header Shipping Information Date Available Hold Pickup Date Fiscal Transaction Information Ext
Pokud je vyžadováno	Item Optional Fields User Optional Fields
Odpověď pokud je problém	Response Header (optional) Problem (R, required) and Required Fee Amount (optional) Required Item Use Restriction Type (R, optional)

	Ext (optional)

Zdroj: vlastní zpracování

Cancel Request	
Použití	Služba požaduje ukončení předchozího požadavku na Item. Zahajující aplikace také může požadovat data o Item nebo User, kterých se relace týká.
Úspěšný výsledek	Odpovídající aplikace ukončí požadavek. Také může poskytnout aktuální údaje o poplatcích User. Dále pošle data pro požadované elementy.
Zprávy a datové elementy	
Cancel Request Initiation Message	
Požadovaná data	User Id or Authentication Input (R)} {Item Id and/or Request Id} Request Type

Zdroj: vlastní zpracování

Příloha 4: Třída LookupItemService:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using System.Xml.Schema;
using System.Xml.Serialization;

namespace DPClient
{
    [Serializable()]
    [XmlRoot(ElementName = "NCIPMessage", Namespace =
"http://www.niso.org/2008/ncip")]
    public class LookupItemService
    {
        [XmlAttribute("version", Namespace =
"http://www.niso.org/2008/ncip", Form = XmlSchemaForm.Qualified)]
        public string version { get; set; }

        private ItemIdE itemId;
        public ItemIdE LookupItem
        {
            get { return itemId; }
            set { itemId = value; }
        }
```

```
    }
    public class ItemIdE
    {
        private ItemIdentifierValueE itemIdentifierValue;
        public ItemIdentifierValueE ItemId
        {
            get { return itemIdentifierValue; }
            set { itemIdentifierValue = value; }
        }
    }
    public class ItemIdentifierValueE
    {
        private string itemId;
        public string ItemIdentifierValue
        {
            get { return itemId; }
            set { itemId = value; }
        }
    }
}
```

Zdroj: vlastní zpracování

Příloha 5: Třída User

```
package org.extensiblecatalog.ncip.v2.dummy;

import java.util.GregorianCalendar;

public class User {
    private String id;
    private String firstname;
    private String lastname;
    private String addressCity;
    private String addressStreetNumber;
    private String addressZipCode;
    private GregorianCalendar dateOfBirth= new GregorianCalendar();

    public void setUserId(String id){
        this.id = id;
    }
    public String getUserId(){
        return this.id;
    }
    public void setUserFirstName(String firstname){
        this.firstname = firstname;
    }
    public String getUserFirstName(){
        return this.firstname;
    }
    public void setUserLastName(String lastname){
        this.lastname = lastname;
    }
    public String getUserLastName(){
        return this.lastname;
    }
    public void setUserAddressCity(String city){
        this.addressCity = city;
```

```
    }
    public String getUserAddressCity()
        return this.addressCity ;
    }
    public void setUserAddressStreetNumber(String streetNumber){
        this.addressStreetNumber= streetNumber;
    }
    public String getUserAddressStreetNumber(){
        return this.addressStreetNumber ;
    }
    public void setUserAddressZipCode(String zipCode){
        this.addressZipCode= zipCode;
    }
    public String getUserAddressZipCode(){
        return this.addressZipCode;
    }

    public void setUserDateOfBirth(String dateOfBirth){
        int year = Integer.parseInt(dateOfBirth.substring(0,
dateOfBirth.indexOf("-")));
        dateOfBirth = dateOfBirth.substring(dateOfBirth.indexOf("-")+1,
dateOfBirth.length());
        int month = Integer.parseInt(dateOfBirth.substring(0,
dateOfBirth.indexOf("-")));
        dateOfBirth = dateOfBirth.substring(dateOfBirth.indexOf("-")+1,
dateOfBirth.length());
        int day = Integer.parseInt(dateOfBirth.substring(0,
dateOfBirth.length()));

        this.dateOfBirth.set(year, month-1, day);
    }
    public GregorianCalendar getUserDateOfBirth(){
        return this.dateOfBirth;
    }
}
```

Zdroj: vlastní zpracování

Příloha 6: DPDatabase

```
package org.extensiblecatalog.ncip.v2.dummy;

import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import org.extensiblecatalog.ncip.v2.service.BibliographicDescription;
import org.extensiblecatalog.ncip.v2.service.ItemId;
import org.extensiblecatalog.ncip.v2.service.PickupLocation;
import org.extensiblecatalog.ncip.v2.service.RequestId;
import org.extensiblecatalog.ncip.v2.service.RequestStatusType;
import org.extensiblecatalog.ncip.v2.service.RequestType;
import org.extensiblecatalog.ncip.v2.service.RequestedItem;
import org.extensiblecatalog.ncip.v2.service.Version1RequestStatusType;
import static org.extensiblecatalog.ncip.v2.service.Version1Request-
StatusType.VERSION_1_REQUEST_STATUS_TYPE;
import org.extensiblecatalog.ncip.v2.service.Version1RequestType;
```

```
import static org.extensiblecatalog.ncip.v2.service.Version1Request-
Type.VERSION_1_REQUEST_TYPE;

public class DPDatabase {
    public static String userName = "sa";
    public static String password = "diplomka";
    public static String url =
"jdbc:sqlserver://localhost\\SQLDP:1433;databaseName=DP";

    public static Item getItem(String itemID) throws SQLException,
ClassNotFoundException
    {
        Item ret = new Item();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "SELECT * FROM [DP].[dbo].[Item] WHERE [id] = " +
itemID;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
           ret.setItemId(rs.getString("id"));
           ret.setItemIdAgency(rs.getString("id_agency"));
           ret.setItemTitle(rs.getString("title"));
           ret.setItemAuthor(rs.getString("author"));
           ret.setItemBibNumber(rs.getString("bib_number"));
           ret.setItemDateCreate(rs.getString("date_of_create"));
           ret.setItemPublisher(rs.getString("publisher"));
           ret.setItemEdition(rs.getString("edition"));
           ret.setItemDatePublish(rs.getString("date_of_publish"));
           ret.setItemLanguage(rs.getString("language"));

ret.setItemStatusOfCirculation(rs.getString("status_of_circulation"));
           ret.setItemCallNumber(rs.getString("call_number"));
           ret.setItemHolding(rs.getString("holding"));
        }
        st.close();
        return ret;
    }
    public static User getUser(String userID) throws SQLException,
ClassNotFoundException
    {
        User ret = new User();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "SELECT * FROM [DP].[dbo].[User] WHERE [id] = " +
userID;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
           ret.setUserId(rs.getString("id"));
           ret.setUserFirstName(rs.getString("firstname"));
           ret.setUserLastName(rs.getString("lastname"));
           ret.setUserAddressCity(rs.getString("address_city"));
```

```
ret.setUserAddressStreetNumber(rs.getString("address_street_number"));
            ret.setUserAddressZipCode(rs.getString("address_zip_code"));
            ret.setUserDateOfBirth(rs.getString("date_of_birth"));
        }
        st.close();
        return ret;
    }
    public static String getItemTitleByUser(String userID, String itemID)
throws SQLException, ClassNotFoundException
    {
        String ret = "";
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "SELECT Item.title FROM Item, Loan WHERE
Loan.id_item = Item.id AND Loan.id_user ="+userID+" AND
Loan.id_item="+itemID;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
            String title = rs.getString("title");
            ret = title;
        }
        st.close();
        return ret;
    }
    public static String getAgencyName(String agencyID)  throws
SQLException, ClassNotFoundException
    {
        String ret = "";
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "SELECT [name] FROM [DP].[dbo].[Agency] WHERE [id] =
" + agencyID;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
            String firstName = rs.getString("name");
            ret = firstName;
        }
        st.close();
        return ret;
    }
    public static String getAgencyCurrency(String agencyID)  throws
SQLException, ClassNotFoundException
    {
        String ret = "";
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "SELECT [currency] FROM [DP].[dbo].[Agency] WHERE
[id] = " + agencyID;
```

```
        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
           String firstName = rs.getString("currency");
           ret = firstName;
        }
        st.close();
        return ret;
    }
    public static ArrayList getLoanedItems(String userID)  throws
SQLException, ClassNotFoundException
    {
        ArrayList<Item> ret = new ArrayList<Item>();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "Select item.title, item.id, Loan.date_to FROM Loan,
Item WHERE id_user="+ userID +" AND Loan.id_item = Item.id" ;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);

        while (rs.next())
        {
           Item item = new Item();
           item.setItemId(rs.getString("id"););
           item.setItemTitle(rs.getString("title"););
           item.setItemDateTo(rs.getString("date_to"););
           ret.add(item);
        }
        st.close();
        return ret;
    }
    public static BigDecimal getAccountBalance(String userID)  throws
SQLException, ClassNotFoundException
    {
        BigDecimal ret = new BigDecimal(0);
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "Select id, account_balance FROM Account WHERE
id_user="+ userID;

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);

        while (rs.next())
        {
           String id = rs.getString("id");
           String accountBalance = rs.getString("account_balance");

           ret = BigDecimal.valueOf(Long.valueOf(accountBalance));
        }
        st.close();
        return ret;
    }
    public static ArrayList getLateFeeList(String userID)  throws
SQLException, ClassNotFoundException
```

```
    {
        ArrayList ret = new ArrayList();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "Select id, id_item, description, value FROM Fee
WHERE id_user="+ userID+" AND name='lateFee'";

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);

        while (rs.next())
        {
           Fee fee = new Fee();
           fee.setFeeDescription(rs.getString("description"));
           fee.setFeeItemId(rs.getString("id_item"));
           fee.setFeeValue(rs.getString("value"));
           ret.add(fee);
        }
        st.close();
        return ret;
    }
    public static ArrayList getIllFeeList(String userID)  throws
SQLException, ClassNotFoundException
    {
        ArrayList ret = new ArrayList();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "Select id, description, value FROM Fee WHERE
id_user="+ userID+" AND name='illFee'";

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);

        while (rs.next())
        {
           Fee fee = new Fee();
           fee.setFeeDescription(rs.getString("description"));
           fee.setFeeId(rs.getString("id"));
           fee.setFeeValue(rs.getString("value"));
           ret.add(fee);
        }
        st.close();
        return ret;
    }
    public static ArrayList getRequestList(String userID)  throws
SQLException, ClassNotFoundException
    {
        ArrayList ret = new ArrayList();
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn = DriverManager.getConnection(url, userName,
password);
        String query = "Select * FROM Request, Item WHERE
id_user="+userID+" AND Request.id_item = Item.id";

        java.sql.Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
```

```
        while (rs.next())
        {
           RequestedItem requestedItem = new RequestedItem();
           RequestId requestId = new RequestId();
           requestId.setRequestIdentifierValue(rs.getString("id"));
           requestedItem.setRequestId(requestId);
           ItemId itemId = new ItemId();
           itemId.setItemIdentifierValue(rs.getString("id_item"));
           requestedItem.setItemId(itemId);

requestedItem.setDatePlaced(toGregorianCalendar(rs.getString("date_create")
));

requestedItem.setPickupDate(toGregorianCalendar(rs.getString("date_pickup_s
tart")));

requestedItem.setPickupExpiryDate(toGregorianCalendar(rs.getString("date_pi
ckup_end")));

           PickupLocation pickupLocation = new
PickupLocation(rs.getString("location_pickup"));
           requestedItem.setPickupLocation(pickupLocation);
           RequestType requestType = new
Version1RequestType(VERSION_1_REQUEST_TYPE, rs.getString("request_type"));
           requestedItem.setRequestType(requestType);
           RequestStatusType requestStatusType = new
Version1RequestStatusType(VERSION_1_REQUEST_STATUS_TYPE,
rs.getString("status_pickup"));
           requestedItem.setRequestStatusType(requestStatusType);
           BibliographicDescription bibDescription = new
BibliographicDescription();
           bibDescription.setTitle(rs.getString("title"));
           ret.add(requestedItem);
        }
        st.close();
        return ret;
    }
    private static GregorianCalendar toGregorianCalendar(String date)
    {
        GregorianCalendar ret = new GregorianCalendar();
        int year = Integer.parseInt(date.substring(0, date.indexOf("-")));
        date = date.substring(date.indexOf("-")+1, date.length());
        int month = Integer.parseInt(date.substring(0, date.indexOf("-")));
        date = date.substring(date.indexOf("-")+1, date.length());
        int day = Integer.parseInt(date.substring(0, date.length()));
        ret.set(year, month-1, day);

        return ret;
    }
}
```

Zdroj: vlastní zpracování

Příloha 7: Třída DummyLookupUserService

```
package org.extensiblecatalog.ncip.v2.dummy;

import java.math.BigDecimal;
```

```
import org.extensiblecatalog.ncip.v2.service.*;

import java.util.*;
public class DummyLookupUserService implements LookupUserService {
    @Override
    public LookupUserResponseData performService(LookupUserInitiationData
initData,
                                                 ServiceContext
serviceContext,
                                                 RemoteServiceManager
serviceManager) {

        final LookupUserResponseData responseData = new
LookupUserResponseData();
        UserOptionalFields userOptionalFields = new UserOptionalFields();
        try
        {
            String agencyidS = "1";
            AgencyId agencyId = new
AgencyId(DPDatabase.getAgencyName(agencyidS));
            Version1CurrencyCode currency = Version1CurrencyCode.CZK;

            responseData.setUserId(initData.getUserId());
            String userNo = initData.getUserId().getUserIdentifierValue();
            ArrayList<Item> loanedItems =
DPDatabase.getLoanedItems(userNo);
            User user = DPDatabase.getUser(userNo);
            if ( initData.getLoanedItemsDesired() ) {
                if(loanedItems.isEmpty() == false){
                    List<LoanedItem> loanedItemsList = new
ArrayList<LoanedItem>();
                    for (Item item : loanedItems ) {
                        LoanedItem loanedItem = new LoanedItem();

                        ItemId itemId = new ItemId();
                        itemId.setItemIdentifierValue(item.getItemId());

                        loanedItem.setItemId(itemId);
                        loanedItem.setDateDue(item.getItemDateTo());

                        Amount amount = new Amount();
                        amount.setCurrencyCode(currency);
                        amount.setMonetaryValue(new BigDecimal(000));
                        loanedItem.setAmount(amount);
                        loanedItem.setRenewalCount(new BigDecimal(0));
                        GregorianCalendar checkoutDate = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                        loanedItem.setDateCheckedOut(checkoutDate);
                        loanedItem.setTitle(item.getItemTitle());
                        loanedItemsList.add(loanedItem);

                        responseData.setLoanedItems(loanedItemsList);
                        List<LoanedItemsCount> loanedItemsCountsList = new
ArrayList<LoanedItemsCount>(1);
                        LoanedItemsCount checkedOutLoanedItemsCount = new
LoanedItemsCount();
                        checkedOutLoanedItemsCount.setCirculationStatus(
```

```
DummyRemoteServiceManager.translateCircStatus(DummyDatabase.CircStatus.CHEC
KED_OUT));

checkedOutLoanedItemsCount.setLoanedItemCountValue(new
BigDecimal(loanedItemsList.size()));

loanedItemsCountsList.add(checkedOutLoanedItemsCount);

responseData.setLoanedItemsCounts(loanedItemsCountsList);
                    }
                }
            }
            if ( initData.getUserFiscalAccountDesired() ) {
                UserFiscalAccount userFiscalAccount = new
UserFiscalAccount();
                List<AccountDetails> accountDetailsList = new
ArrayList<AccountDetails>();
                List<UserFiscalAccount> userFiscalAccountsList = new
ArrayList<UserFiscalAccount>();
                userFiscalAccount.setAccountDetails(accountDetailsList);
                userFiscalAccountsList.add(userFiscalAccount);
                responseData.setUserFiscalAccounts(userFiscalAccountsList);

                ArrayList lateFeeList = DPDatabase.getLateFeeList(userNo);
                for(int i = 0; i < lateFeeList.size(); i++){
                    Fee fee = (Fee)lateFeeList.get(i);

                    AccountDetails lateFeeAccountDetails = new
AccountDetails();
                    GregorianCalendar lateFeeAccrualDate = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    lateFeeAccrualDate.add(Calendar.DAY_OF_YEAR, -45);

lateFeeAccountDetails.setAccrualDate(lateFeeAccrualDate);
                    accountDetailsList.add(lateFeeAccountDetails);

                    ItemId lateFeeItemId = new ItemId();

lateFeeItemId.setItemIdentifierValue(fee.getFeeItemId());
                    FiscalTransactionInformation
lateFeeFiscalTransactionInformation = new FiscalTransactionInformation();

lateFeeAccountDetails.setFiscalTransactionInformation(lateFeeFiscalTransact
ionInformation);

                    Amount lateFeeAmount = new Amount();
                    lateFeeAmount.setCurrencyCode(currency);
                    lateFeeAmount.setMonetaryValue(new
BigDecimal(fee.getFeeValue()));


lateFeeFiscalTransactionInformation.setAmount(lateFeeAmount);

lateFeeFiscalTransactionInformation.setFiscalActionType(Version1FiscalActio
nType.ASSESS);

lateFeeFiscalTransactionInformation.setFiscalTransactionDescription(fee.get
FeeDescription());
```

```
                    FiscalTransactionReferenceId
lateFeeFiscalTransactionReferenceId = new FiscalTransactionReferenceId();

lateFeeFiscalTransactionReferenceId.setFiscalTransactionIdentifierValue("la
teFee");

lateFeeFiscalTransactionReferenceId.setAgencyId(agencyId);

lateFeeFiscalTransactionInformation.setFiscalTransactionReferenceId(lateFee
FiscalTransactionReferenceId);

lateFeeFiscalTransactionInformation.setFiscalTransactionType(Version1Fiscal
TransactionType.FINE);

                    ItemDetails lateFeeItemDetails = new ItemDetails();
                    BibliographicDescription lateFeeBibDescription = new
BibliographicDescription();

lateFeeBibDescription.setTitle(DPDatabase.getItemTitleByUser(userNo,
fee.getFeeItemId()));

lateFeeItemDetails.setBibliographicDescription(lateFeeBibDescription);

                    GregorianCalendar lateFeeCheckoutDate = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    lateFeeCheckoutDate.add(Calendar.DAY_OF_YEAR, -75);

lateFeeItemDetails.setDateCheckedOut(lateFeeCheckoutDate);
                    GregorianCalendar lateFeeDateDue = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    lateFeeDateDue.add(Calendar.DAY_OF_YEAR, -50);
                    lateFeeItemDetails.setDateDue(lateFeeDateDue);
                    GregorianCalendar lateFeeDateReturned = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    lateFeeDateReturned.add(Calendar.DAY_OF_YEAR, -45);

lateFeeItemDetails.setDateReturned(lateFeeDateReturned);
                    lateFeeItemDetails.setItemId(lateFeeItemId);

lateFeeFiscalTransactionInformation.setItemDetails(lateFeeItemDetails);

                    GregorianCalendar lateFeeValidFromDate = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    lateFeeValidFromDate.add(Calendar.DAY_OF_YEAR, -45);

lateFeeFiscalTransactionInformation.setValidFromDate(lateFeeValidFromDate);
                    GregorianCalendar lateFeeValidToDate = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    lateFeeValidToDate.add(Calendar.YEAR, 1);

lateFeeFiscalTransactionInformation.setValidToDate(lateFeeValidToDate);
                }
                ArrayList illFeeList = DPDatabase.getIllFeeList(userNo);

                for(int i = 0; i < illFeeList.size(); i++){
                    Fee fee = (Fee)illFeeList.get(i);
                    AccountDetails illFineAccountDetails = new
AccountDetails();
```

```
                    GregorianCalendar illFineAccrualDate = new
GregorianCalendar(TimeZone.getTimeZone("UTC"));
                    illFineAccrualDate.add(Calendar.DAY_OF_YEAR, -5);

illFineAccountDetails.setAccrualDate(illFineAccrualDate);
                    accountDetailsList.add(illFineAccountDetails);

                    FiscalTransactionInformation
illFineFiscalTransactionInformation = new FiscalTransactionInformation();

illFineAccountDetails.setFiscalTransactionInformation(illFineFiscalTransact
ionInformation);

                    Amount illFineAmount = new Amount();
                    illFineAmount.setCurrencyCode(currency);
                    illFineAmount.setMonetaryValue(new
BigDecimal(fee.getFeeValue()));

illFineFiscalTransactionInformation.setAmount(illFineAmount);

illFineFiscalTransactionInformation.setFiscalActionType(Version1FiscalActio
nType.ASSESS);

illFineFiscalTransactionInformation.setFiscalTransactionDescription(fee.get
FeeDescription());
                    FiscalTransactionReferenceId
illFineFiscalTransactionReferenceId = new FiscalTransactionReferenceId();

illFineFiscalTransactionReferenceId.setFiscalTransactionIdentifierValue("il
lFee");

illFineFiscalTransactionReferenceId.setAgencyId(agencyId);

illFineFiscalTransactionInformation.setFiscalTransactionReferenceId(illFine
FiscalTransactionReferenceId);

illFineFiscalTransactionInformation.setFiscalTransactionType(Version1Fiscal
TransactionType.INTERLIBRARY_LOAN_FEE);
                    RequestId illFineRequestId = new RequestId();
                    illFineRequestId.setRequestIdentifierValue("");

illFineFiscalTransactionInformation.setRequestId(illFineRequestId);
                }
                AccountBalance accountBalance = new AccountBalance();
                accountBalance.setCurrencyCode(currency);

accountBalance.setMonetaryValue(DPDatabase.getAccountBalance(userNo));
                userFiscalAccount.setAccountBalance(accountBalance);

                UserFiscalAccountSummary userFiscalAcctSummary = new
UserFiscalAccountSummary();
                userFiscalAcctSummary.setAccountBalance(accountBalance);
                userFiscalAcctSummary.setChargesCount(new
BigDecimal(accountDetailsList.size()));

responseData.setUserFiscalAccountSummary(userFiscalAcctSummary);
            }
            if ( initData.getRequestedItemsDesired() ) {
```

```
                List<RequestedItem> requestedItemsList =
DPDatabase.getRequestList(userNo);
                Map<CirculationStatus, BigDecimal> countByCircStatus
                    = new HashMap<CirculationStatus, BigDecimal>();
                responseData.setRequestedItems(requestedItemsList);

                if ( countByCircStatus.size() > 0 ) {
                    List<RequestedItemsCount> requestedItemsCountsList
                        = new
ArrayList<RequestedItemsCount>(countByCircStatus.size());

                    for ( Map.Entry<CirculationStatus, BigDecimal> entry :
countByCircStatus.entrySet() ) {
                        RequestedItemsCount requestedItemsCount = new
RequestedItemsCount();

requestedItemsCount.setCirculationStatus(entry.getKey());

requestedItemsCount.setRequestedItemCountValue(entry.getValue());
                        requestedItemsCountsList.add(requestedItemsCount);
                    }

responseData.setRequestedItemsCounts(requestedItemsCountsList);

                    CirculationStatus ncipCircStatus =
DummyRemoteServiceManager.translateCircStatus(DummyDatabase.CircStatus.ON_S
HELF);
                    BigDecimal count =
countByCircStatus.get(ncipCircStatus);
                    if ( count != null ) {
                        count = count.add(new BigDecimal(1));
                    } else {
                        count = new BigDecimal(1);
                    }
                    countByCircStatus.put(ncipCircStatus, count);
                }
                responseData.setRequestedItems(requestedItemsList);

                if ( countByCircStatus.size() > 0 ) {

                    List<RequestedItemsCount> requestedItemsCountsList
                        = new
ArrayList<RequestedItemsCount>(countByCircStatus.size());

                    for ( Map.Entry<CirculationStatus, BigDecimal> entry :
countByCircStatus.entrySet() ) {

                        RequestedItemsCount requestedItemsCount = new
RequestedItemsCount();

requestedItemsCount.setCirculationStatus(entry.getKey());

requestedItemsCount.setRequestedItemCountValue(entry.getValue());
                        requestedItemsCountsList.add(requestedItemsCount);

                    }

responseData.setRequestedItemsCounts(requestedItemsCountsList);
                }
```

```
            initData.setDateOfBirthDesired(true);
            if ( initData.getDateOfBirthDesired()) {

userOptionalFields.setDateOfBirth(user.getUserDateOfBirth());
            }
            initData.setNameInformationDesired(true);
            if (initData.getNameInformationDesired()) {
            }
                PersonalNameInformation pni = new
PersonalNameInformation();

pni.setUnstructuredPersonalUserName(user.getUserFirstName()+ " " +
user.getUserLastName());

                NameInformation ni = new NameInformation();
                ni.setPersonalNameInformation(pni);
                userOptionalFields.setNameInformation(ni);
            }
            responseData.setUserOptionalFields(userOptionalFields);
        }catch(Exception e)
        {
            System.out.println(e);
        }
        return responseData;
    }
}
```

Zdroj: vlastní zpracování

Příloha 8: Třída DummyLookupItemService

```
package org.extensiblecatalog.ncip.v2.dummy;

import org.extensiblecatalog.ncip.v2.service.*;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

public class DummyLookupItemService implements LookupItemService {

    @Override
    public LookupItemResponseData performService(LookupItemInitiationData
initData,
                                                 ServiceContext
serviceContext,
                                                 RemoteServiceManager
serviceManager) throws ServiceException {

        final LookupItemResponseData responseData = new
LookupItemResponseData();
        try
        {
            BibliographicDescription bibDesc = new
BibliographicDescription();

            List<Problem> problems = null;
            if ( initData.getItemId() != null ) {
                if (initData.getItemId().getItemIdentifierValue() == null){
                    problems =
ServiceHelper.generateProblems(Version1LookupItemProcessingError.UNKNOWN_IT
EM, "LookupItem",
                        null, "Item " +
initData.getItemId().getItemIdentifierValue() + " nenalezen.");
                }
            } else if ( initData.getRequestId() != null ) {
                if ( initData.getRequestId().getRequestIdentifierValue() !=
null ){
                    if (initData.getItemId().getItemIdentifierValue() ==
null){
                        problems =
ServiceHelper.generateProblems(Version1LookupItemProcessingError.UNKNOWN_IT
EM, "LookupItem",
                            null, "Item " +
initData.getItemId().getItemIdentifierValue() + " nenalezen.");
                    }
                }
                else{
                    problems =
ServiceHelper.generateProblems(Version1LookupItemProcessingError.UNKNOWN_IT
EM, "LookupItem",
                        null, "Request " +
initData.getRequestId().getRequestIdentifierValue() + " nenalezen.");
                }
            } else {
                problems =
ServiceHelper.generateProblems(Version1GeneralProcessingError.NEEDED_DATA_M
ISSING, "LookupItem",
                    null, "Buď ItemId nebo RequestId je potřeba.");
```

```
            }
            Item item = new Item();
            try{
                item =
DPDatabase.getItem(initData.getItemId().getItemIdentifierValue());
            }catch(Exception e){
                problems =
ServiceHelper.generateProblems(Version1LookupItemProcessingError.UNKNOWN_IT
EM, "LookupItem",
                        null, "Item " +
initData.getItemId().getItemIdentifierValue() + " nenalezen.");
            }
            if (initData.getItemId().getItemIdentifierValue() != null) {
                ItemId itemId = new ItemId();
                itemId.setItemIdentifierValue(item.getItemId());
                responseData.setItemId(itemId);
                if ( initData.getRequestId() != null ){
                    responseData.setRequestId(initData.getRequestId());
                }
                BibliographicRecordId bibliographicRecordId = new
BibliographicRecordId();

bibliographicRecordId.setBibliographicRecordIdentifier(item.getItemBibNumbe
r());
                AgencyId agencyId = new
AgencyId(DPDatabase.getAgencyName(item.getItemIdAgency()));
                bibliographicRecordId.setAgencyId(agencyId);
                List<BibliographicRecordId> bibRecordIds = new
ArrayList<BibliographicRecordId>();
                bibRecordIds.add(bibliographicRecordId);
                bibDesc.setBibliographicRecordIds(bibRecordIds);

                Language language =
Language.find(Version1Language.VERSION_1_LANGUAGE, item.getItemLanguage());
                bibDesc.setLanguage(language);
                bibDesc.setAuthor(item.getItemAuthor());
                bibDesc.setEdition(item.getItemEdition());
                bibDesc.setPublisher(item.getItemPublisher());
                bibDesc.setPublicationDate(item.getItemDatePublish());
                bibDesc.setTitle(item.getItemTitle());

                CirculationStatus circStatus =
Version1CirculationStatus.CIRCULATION_STATUS_UNDEFINED;
                if(item.getItemStatusOfCirculation().equals("ON_ORDER")){
                    circStatus = Version1CirculationStatus.IN_PROCESS;
                }
                if(item.getItemStatusOfCirculation().equals("ON_SHELF")){
                    circStatus =
Version1CirculationStatus.AVAILABLE_ON_SHELF;

}if(item.getItemStatusOfCirculation().equals("CHECKED_OUT")){
                    circStatus = Version1CirculationStatus.ON_LOAN;
                }
if(item.getItemStatusOfCirculation().equals("IN_TRANSIT")){
                    circStatus =
Version1CirculationStatus.IN_TRANSIT_BETWEEN_LIBRARY_LOCATIONS;
                }
                ItemDescription itemDescription = new ItemDescription();
                itemDescription.setCallNumber(item.getItemCallNumber());
```

```
                if ( item.getItemHolding() != null ) {
                    HoldingsInformation holdingsInfo = new
HoldingsInformation();

holdingsInfo.setUnstructuredHoldingsData(item.getItemHolding());
                    itemDescription.setHoldingsInformation(holdingsInfo);
                }
                itemDescription.setNumberOfPieces(new BigDecimal(1));
                ItemOptionalFields itemOptionalFields = new
ItemOptionalFields();
                itemOptionalFields.setBibliographicDescription(bibDesc);
                itemOptionalFields.setCirculationStatus(circStatus);
                itemOptionalFields.setItemDescription(itemDescription);

                responseData.setItemOptionalFields(itemOptionalFields);
            } else if ( problems == null ) {
                problems =
ServiceHelper.generateProblems(Version1GeneralProcessingError.TEMPORARY_PRO
CESSING_FAILURE,
                    "LookupItem", null, "Neznámá chyba.");
                responseData.setProblems(problems);
            } else {
                responseData.setProblems(problems);
            }
        }
        catch(Exception e){}
        return responseData;
    }
}
```

Zdroj: vlastní zpracování

Příloha 9: Testovací zpráva – odpověď na Lookup User

```
<ns1:NCIPMessage xmlns:ns1="http://www.niso.org/2008/ncip"
xmlns:ns2="http://oclc.org/WCL/ncip/2011/extensions"
xmlns:ns3="http://www.oclc.org/ncip/usernote/2012"
ns1:version="http://www.niso.org/schemas/ncip/v2_0/imp1/xsd/ncip_v2_0.xsd">
 <ns1:LookupUserResponse>
  <ns1:UserId>
   <ns1:UserIdentifierValue>2</ns1:UserIdentifierValue>
  </ns1:UserId>
  <ns1:UserFiscalAccount>
   <ns1:AccountBalance>
    <ns1:CurrencyCode ns1:Scheme="http://www.bsi-global.com/Technical+Infor-
mation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
    <ns1:MonetaryValue>1535</ns1:MonetaryValue>
   </ns1:AccountBalance>
   <ns1:AccountDetails>
    <ns1:AccrualDate>2016-01-23T12:40:17.408Z</ns1:AccrualDate>
    <ns1:FiscalTransactionInformation>
```

```
        <ns1:FiscalActionType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscalaction-
type/fiscalactiontype.scm">Assess</ns1:FiscalActionType>
        <ns1:FiscalTransactionReferenceId>
         <ns1:AgencyId>Knihovna ČB</ns1:AgencyId>
         <ns1:FiscalTransactionIdentifierValue>lateFee</ns1:FiscalTransactionIdentifi-
erValue>
        </ns1:FiscalTransactionReferenceId>
        <ns1:FiscalTransactionType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscaltransaction-
type/fiscaltransactiontype.scm">Fine</ns1:FiscalTransactionType>
        <ns1:ValidFromDate>2016-01-23T12:40:17.434Z</ns1:ValidFromDate>
        <ns1:ValidToDate>2017-03-08T12:40:17.434Z</ns1:ValidToDate>
        <ns1:Amount>
         <ns1:CurrencyCode     ns1:Scheme="http://www.bsi-global.com/Technical+In-
formation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
         <ns1:MonetaryValue>50</ns1:MonetaryValue>
        </ns1:Amount>
        <ns1:FiscalTransactionDescription>poplatek za pozdní vrácení</ns1:FiscalTrans-
actionDescription>
        <ns1:ItemDetails>
         <ns1:ItemId>
          <ns1:ItemIdentifierValue>3</ns1:ItemIdentifierValue>
         </ns1:ItemId>
         <ns1:BibliographicDescription>
          <ns1:Author>Karel Čapek</ns1:Author>
          <ns1:Title>Krakatit</ns1:Title>
         </ns1:BibliographicDescription>
         <ns1:DateCheckedOut>2015-12-24T12:40:17.434Z</ns1:DateCheckedOut>
         <ns1:DateDue>2016-01-18T12:40:17.434Z</ns1:DateDue>
         <ns1:DateReturned>2016-01-23T12:40:17.434Z</ns1:DateReturned>
        </ns1:ItemDetails>
       </ns1:FiscalTransactionInformation>
     </ns1:AccountDetails>
     <ns1:AccountDetails>
      <ns1:AccrualDate>2016-01-23T12:40:17.434Z</ns1:AccrualDate>
      <ns1:FiscalTransactionInformation>
       <ns1:FiscalActionType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscalaction-
type/fiscalactiontype.scm">Assess</ns1:FiscalActionType>
        <ns1:FiscalTransactionReferenceId>
         <ns1:AgencyId>Knihovna ČB</ns1:AgencyId>
         <ns1:FiscalTransactionIdentifierValue>lateFee</ns1:FiscalTransactionIdentifi-
erValue>
        </ns1:FiscalTransactionReferenceId>
```

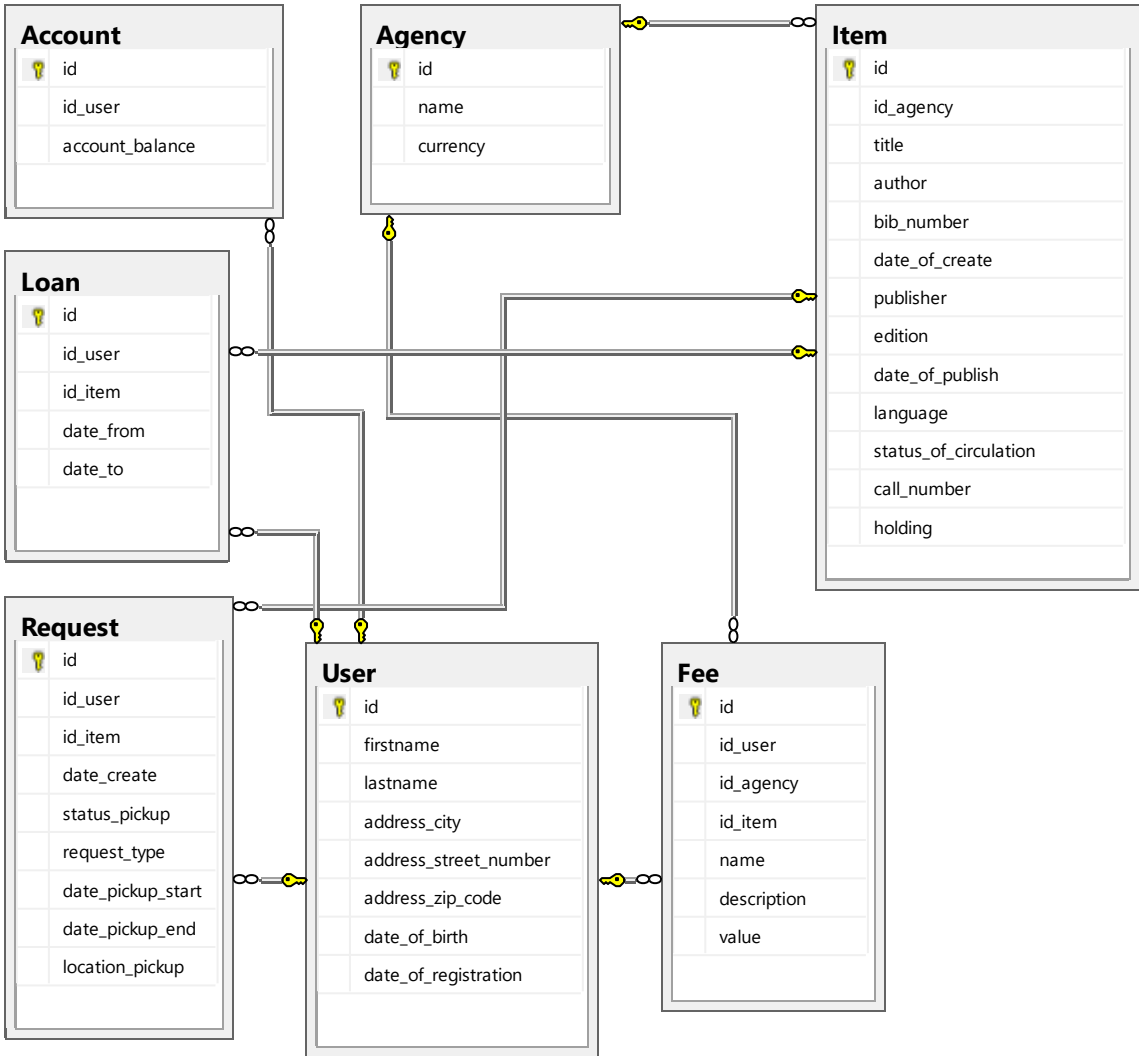
```
        <ns1:FiscalTransactionType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscaltransaction-
type/fiscaltransactiontype.scm">Fine</ns1:FiscalTransactionType>
        <ns1:ValidFromDate>2016-01-23T12:40:17.461Z</ns1:ValidFromDate>
        <ns1:ValidToDate>2017-03-08T12:40:17.461Z</ns1:ValidToDate>
        <ns1:Amount>
         <ns1:CurrencyCode     ns1:Scheme="http://www.bsi-global.com/Technical+In-
formation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
         <ns1:MonetaryValue>80</ns1:MonetaryValue>
        </ns1:Amount>
        <ns1:FiscalTransactionDescription>poplatek za pozdní vrácení</ns1:FiscalTrans-
actionDescription>
        <ns1:ItemDetails>
         <ns1:ItemId>
          <ns1:ItemIdentifierValue>2</ns1:ItemIdentifierValue>
         </ns1:ItemId>
         <ns1:BibliographicDescription>
          <ns1:Author>Karel Hynek Mácha</ns1:Author>
          <ns1:Title>Máj</ns1:Title>
         </ns1:BibliographicDescription>
         <ns1:DateCheckedOut>2015-12-24T12:40:17.461Z</ns1:DateCheckedOut>
         <ns1:DateDue>2016-01-18T12:40:17.461Z</ns1:DateDue>
         <ns1:DateReturned>2016-01-23T12:40:17.461Z</ns1:DateReturned>
        </ns1:ItemDetails>
       </ns1:FiscalTransactionInformation>
     </ns1:AccountDetails>
     <ns1:AccountDetails>
      <ns1:AccrualDate>2016-03-03T12:40:17.473Z</ns1:AccrualDate>
      <ns1:FiscalTransactionInformation>
       <ns1:FiscalActionType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscalaction-
type/fiscalactiontype.scm">Assess</ns1:FiscalActionType>
        <ns1:FiscalTransactionReferenceId>
         <ns1:AgencyId>Knihovna ČB</ns1:AgencyId>
         <ns1:FiscalTransactionIdentifierValue>illFee</ns1:FiscalTransactionIdentifier-
Value>
        </ns1:FiscalTransactionReferenceId>
        <ns1:FiscalTransactionType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/fiscaltransaction-
type/fiscaltransactiontype.scm">Interlibrary Loan Fee</ns1:FiscalTransactionType>
        <ns1:Amount>
         <ns1:CurrencyCode     ns1:Scheme="http://www.bsi-global.com/Technical+In-
formation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
         <ns1:MonetaryValue>300</ns1:MonetaryValue>
        </ns1:Amount>
        <ns1:FiscalTransactionDescription>poplatek     za     ILL</ns1:FiscalTransaction-
Description>
```

```
      <ns1:RequestId>
       <ns1:RequestIdentifierValue></ns1:RequestIdentifierValue>
      </ns1:RequestId>
     </ns1:FiscalTransactionInformation>
    </ns1:AccountDetails>
   </ns1:UserFiscalAccount>
   <ns1:LoanedItemsCount>
    <ns1:CirculationStatus
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/circulationstatus/cir-
culationstatus.scm">On Loan</ns1:CirculationStatus>
    <ns1:LoanedItemCountValue>2</ns1:LoanedItemCountValue>
   </ns1:LoanedItemsCount>
   <ns1:LoanedItem>
    <ns1:ItemId>
     <ns1:ItemIdentifierValue>1</ns1:ItemIdentifierValue>
    </ns1:ItemId>
    <ns1:DateDue>2016-06-01T11:40:17.353Z</ns1:DateDue>
    <ns1:Amount>
     <ns1:CurrencyCode    ns1:Scheme="http://www.bsi-global.com/Technical+Infor-
mation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
     <ns1:MonetaryValue>0</ns1:MonetaryValue>
    </ns1:Amount>
    <ns1:Title>Máj</ns1:Title>
    <ns1:Ext>
     <ns1:RenewalCount>0</ns1:RenewalCount>
     <ns1:DateCheckedOut>2016-03-08T12:40:17.377Z</ns1:DateCheckedOut>
    </ns1:Ext>
   </ns1:LoanedItem>
   <ns1:LoanedItem>
    <ns1:ItemId>
     <ns1:ItemIdentifierValue>3</ns1:ItemIdentifierValue>
    </ns1:ItemId>
    <ns1:DateDue>2016-06-01T11:40:17.353Z</ns1:DateDue>
    <ns1:Amount>
     <ns1:CurrencyCode    ns1:Scheme="http://www.bsi-global.com/Technical+Infor-
mation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
     <ns1:MonetaryValue>0</ns1:MonetaryValue>
    </ns1:Amount>
    <ns1:Title>Krakatit</ns1:Title>
    <ns1:Ext>
     <ns1:RenewalCount>0</ns1:RenewalCount>
     <ns1:DateCheckedOut>2016-03-08T12:40:17.382Z</ns1:DateCheckedOut>
    </ns1:Ext>
   </ns1:LoanedItem>
   <ns1:RequestedItem>
    <ns1:RequestId>
     <ns1:RequestIdentifierValue>1</ns1:RequestIdentifierValue>
```

```
    </ns1:RequestId>
    <ns1:ItemId>
     <ns1:ItemIdentifierValue>1</ns1:ItemIdentifierValue>
    </ns1:ItemId>
    <ns1:RequestType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/requesttype/request-
type.scm">Loan</ns1:RequestType>
    <ns1:RequestStatusType
ns1:Scheme="http://www.niso.org/ncip/v1_0/imp1/schemes/requeststatus-
type/requeststatustype.scm">Available For Pickup</ns1:RequestStatusType>
    <ns1:DatePlaced>2016-02-01T12:40:17.500Z</ns1:DatePlaced>
    <ns1:PickupDate>2016-04-01T11:40:17.500Z</ns1:PickupDate>
    <ns1:PickupLocation>Knihovna ČB</ns1:PickupLocation>
    <ns1:PickupExpiryDate>2016-06-01T11:40:17.500Z</ns1:PickupExpiryDate>
    <ns1:Title>Babička</ns1:Title>
   </ns1:RequestedItem>
   <ns1:UserOptionalFields>
    <ns1:NameInformation>
     <ns1:PersonalNameInformation>
      <ns1:UnstructuredPersonalUserName>Jakub  Devátý</ns1:UnstructuredPerso-
nalUserName>
     </ns1:PersonalNameInformation>
    </ns1:NameInformation>
    <ns1:DateOfBirth>1991-01-01T12:40:17.357Z</ns1:DateOfBirth>
   </ns1:UserOptionalFields>
   <ns1:Ext>
    <ns2:UserFiscalAccountSummary>
     <ns2:ChargesCount>3</ns2:ChargesCount>
     <ns1:AccountBalance>
      <ns1:CurrencyCode  ns1:Scheme="http://www.bsi-global.com/Technical+Infor-
mation/Publications/_Publications/tig90x.doc">CZK</ns1:CurrencyCode>
      <ns1:MonetaryValue>1535</ns1:MonetaryValue>
     </ns1:AccountBalance>
    </ns2:UserFiscalAccountSummary>
   </ns1:Ext>
  </ns1:LookupUserResponse>
</ns1:NCIPMessage>
```

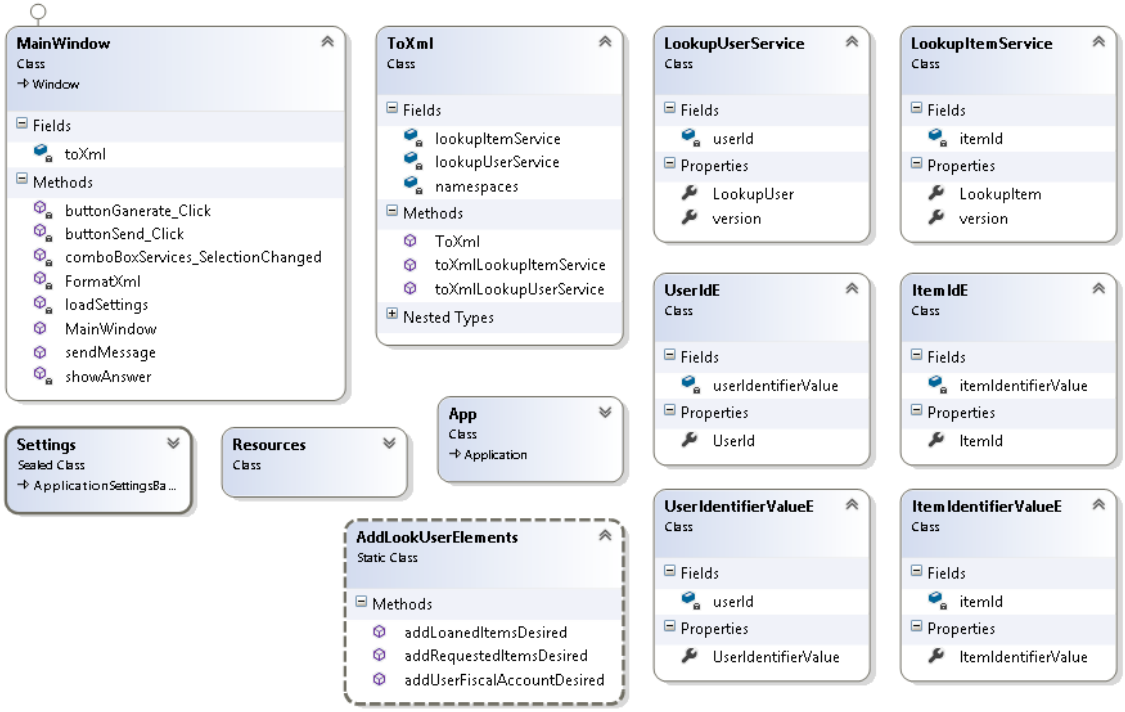
Zdroj: vlastní zpracování

Příloha 10: Diagram databáze



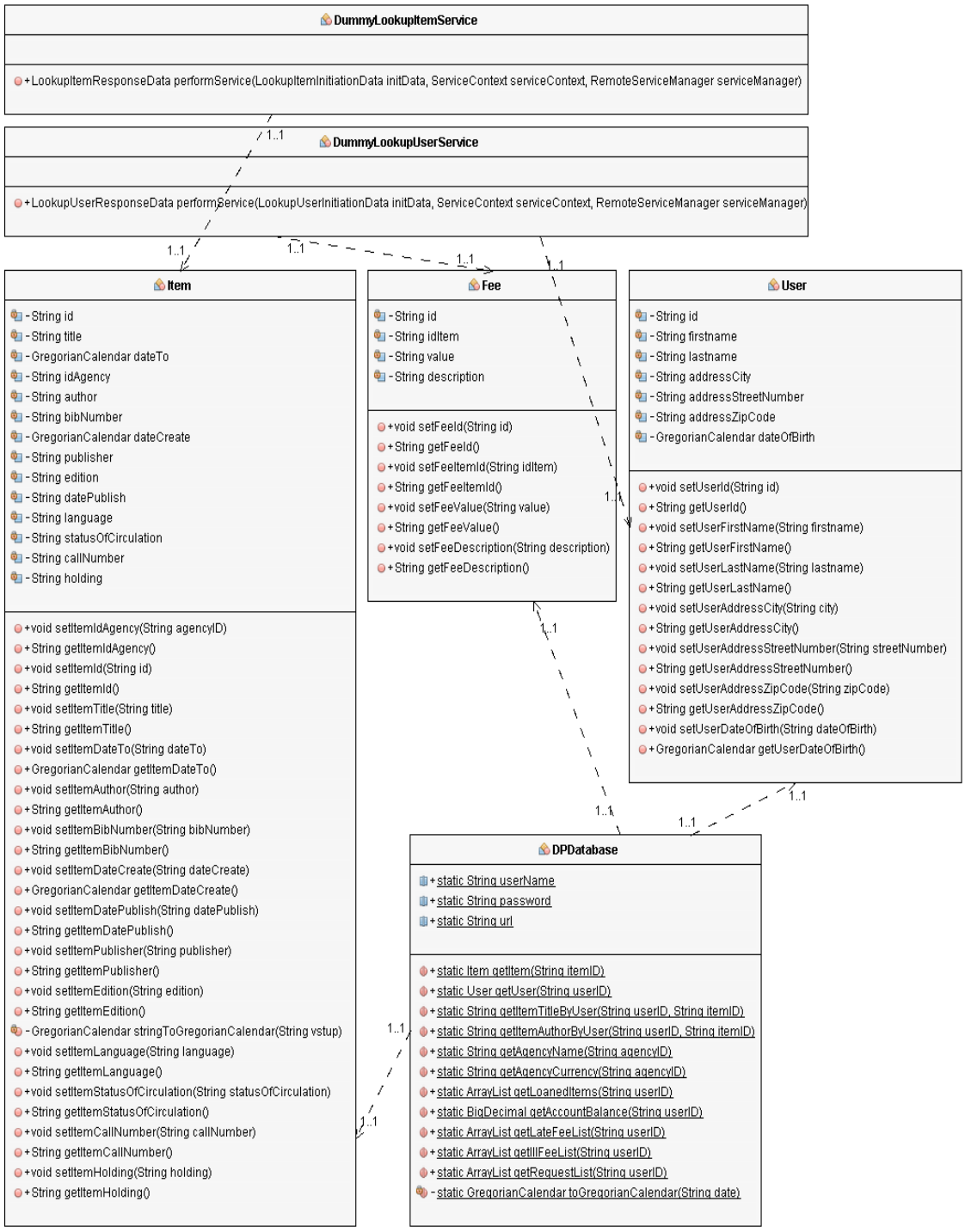
Zdroj: vlastní zpracování

Příloha 11: Diagram tříd - Initiator



Zdroj: vlastní zpracování

Příloha 12: Diagram tříd - Responder



Zdroj: vlastní zpracování

Příloha 13: CD s projekty aplikací a databází