БИСИС

Бојана Димић
Душан Сурла

Обрада библиографске грађе у софтверском систему БИСИС

Иновациони центар за електронске библиотеке и архиве

Департман за математику и информатику

Природно–математички факултет

Универзитет у Новом Саду

Бојана Димић

Душан Сурла

Обрада библиографске грађе у софтверском систему БИСИС

Нови Сад, 2007.

Назив монографије

Обрада библиографске грађе у софтверском систему БИСИС

Аутори

мр Бојана Димић , истраживач приправник, Иновациони центар за електронске библиотеке и архиве, Департман за математику и информатику, Природно-математички факултет, Нови Сад

др Душан Сурла, редовни професор, Иновациони центар за електронске библиотеке и архиве, Департман за математику и информатику, Природно-математички факултет, Нови Сад

Рецензенти

др Зора Коњовић, редовни професор, Факултет техничких наука, Нови Сад

др Милош Рацковић, редовни професор, Природно-математички факултет, Нови Сад

др Бранко Милосављевић, редовни професор, Факултет техничких наука, Нови Сад

Издавач

Природно-математички факултет, Нови Сад

Главни и одговорни уредник

Проф. др Мирослав Весковић, декан Природно-математичког факултета у Новом Саду

Нучно-наставно веће Природно-математичког факултета у Новом Саду, на седници 22. 10. 2007. године, одобрило је штампање и употребу ове монографије.

Корице

Данијела Тешендић

Штампа

АМБ Економик

Тираж

150

Садржај

Предговор

У овој монографији приказан је део добијених резултата на пројекту *Апстрактни модели и примене у рачунарским наукама*, број 144017А, који финансира Министарство науке и заштите животне средине Републике Србије. Носилац истраживања овог пројекта је Природно-математички факултет у Новом Саду.

Монографија припада области пројектовања библиотечких информационих система. Приказано је моделирање и имплементација едитора за обраду библиографске грађе по YUMARC формату, који представља варијанту UNIMARC формата прилагођену за националну употребу и коришћење у библиотечком софтверском систему БИСИС. Едитор је реализован у Java окружењу (Swing компоненте) коришћењем XML технологија.

Монографија садржи следећа поглавља:

1. Увод
2. XML шеме YUMARC формата и библиографског записа по YUMARC формату
3. Моделирање XML едитора за обраду библиографске грађе
4. Имплементација XML едитора за обраду библиографске грађе
5. Опис коришћења едитора за обраду библиографске грађе

У **првом** поглављу је дат опис развоја библиотечког софтверског система БИСИС као и опис функционалности едитора за обраду библиографске грађе и преглед постојећих решења из ове области.

У **другом** поглављу је дат опис основних концепата библиотечког формата YUMARC, модел XML докумената YUMARC формата и XML библиографских записа по YUMARC формату. Моделирање је извршено у XMLSchema језику.

Моделирање система за обраду библиографске грађе дато је у **трећем** поглављу. Ово поглавље садржи спецификацију информационих захтева система путем описа случајева коришћења. Дат је и статички модел система путем дијаграм класа, као и динамички дијаграми активности.

У **четвртом** поглављу описано је коришћено софтверско окружење као и основни концепти имплементације XML едитора за обраду библиографске грађе.

Опис коришћења XML едитора за обраду библиографске грађе у оквиру четврте верзије библиотечког софтверског система БИСИС дат је у **петом** поглављу.

Захваљујемо се рецензентима на корисним примедбама и сугестијама које су допринеле квалитету монографије.

Нови Сад, 2007. *Аутори*

Сажетак

Циљ. Циљ истраживања је моделирање и имплементација XML едитора за обраду библиографске грађе у библиотечком софтверском систему БИСИС. Моделирање информационих захтева овог едитора урађено је у обједињеном језику моделирања (UML 2.0). Имплементација система реализована је у програмском језику Java.

Методологија. Коришћена је објектно оријентисана методологија за пројектовање софтверских компоненти као и CASE алати. Архитектура софтверског система подржава рад са различитим библиотечким стандардима и може се интегрисати у различите библиотечке софтверске системе.

Резултат истраживања. Резултат је апликација едитора за обраду библиографске грађе по YUMARC формату. Имплементација едитора је заснована на XML технологијама чиме су остваране две основне карактеристике, а то су могућност интеграције едитора у различите библиотечке софтверске системе и да је за прелазак на други формат за обраду библиогарфске грађе потребно изменити само модул за контролу података библиогарфских записа.

Ограничење резултата истраживања. Основно ограничење система односи се на подсистем који врши контролу библиогарфских записа и његово проширење за рад са другим библиографским форматима. У предложеном решењу део контроле уноса података укључен је у саму имплементацију и односи се на библиогарфски формат YUMARC. Наиме, део података по којима се врши контрола, као што су поновљивост елемената записа и шифарници налази се у XML документу формата који је улазна информација у едитору. Међутим, контрола која се односи на валидацију формата садржаја елемената записа не може се извршити за неки други формат без промене у имплементацији. Према томе, истраживање би се могло наставити у правцу разматрања могућности да се подаци по којима се врши контрола садржаја издвоје као улазна информација за апликацију, чиме би и овај део био независан од имплементације. Једна од могућности би била проширење XML документа формата овим подацима. Неко друго решење би подразумевало креирање потпуно одвојеног система за валидацију садржаја.

*Практична примена***.** Систем за обраду библиографске грађе интегрисан је у четврту верзију система БИСИС. Тестирање и верификација је извршена над библиографским записима јавних градских библиотека. Систем БИСИС вер. 3 ради у 36 библиотека (факултетске, јавне и специјализоване). У току је прелазак ових библиотека на четврту верзију система.

*Оригиналност резултата истраживања***.** Оригиналност рада је у архитектури система која је заснована на XML документима и независна од библиотечког формата и софтверског система у који се систем интегрише. XML документ који садржи податке о библиотечком формату представља улазну информацију у едитору. Према томе, увођење новог формата за обраду библиографске грађе састоји су у креирању одговарајућег XML документа који би се као улазна информација проследио едитору. Након креирања библиографског записа у овом едитору, запис се складишти у XML документ који представља излазну информацију из едитора. Овај XML документ може се складиштити у различите софтверске системе за чување и претраживање података.

Abstract

Purpose. The purpose of the research is modelling and implementation of XML editor for bibliographic material processing in library software system BISIS. Information requirements of the editor are modelled in Unified Modelling Language (UML 2.0). System implementation is realised in Java programming language.

Methodology. The object-oriented methodology as well as CASE tools is used to design the software components. The software architecture supports work with different bibliographic standards and can be integrated into various library software systems.

Findings. The outcome is application for processing bibliographic material by YUMARC format. The implementation of the editor is based on XML technology. XML technology enables two basic characteristics of the system and these are the possibility of integrating system into different library software systems and that only changes in module for control of bibliographic records data are required for moving to another format for bibliographic processing.

Research limitations. The limitation of the system relates to the subsystem for bibliographic records control and its extension for other bibliographic formats. In proposed solution, the part of the data entry control is included into implementation and is related only to YUMARC format. Namely, the part of data for control, such as the possibility of elements to be repeated and coders are placed in XML document which is the input parameter. Still, the control of the format of record data cannot be realised without changes in implementation. Nevertheless, the subject for further research is the system in which these data will be input parameter, as well. In that way, this part of control will also be separate from implementation. One suggestion for this is extension of XML document with these data. Another suggestion is developing completely separated system for content control.

Practical implications. The system for bibliographic material processing is integrated into forth version of system BISIS. Its testing and verification were done by using bibliographic records from public city libraries. The system BISIS, version 3 is in use in 36 libraries (faculty, public and specialised). The installation of forth version of system BISIS is in progress.

The originality of research findings. The originality lies in system architecture which is based on XML documents and independent of bibliographic format. Also, the system can be integrated into various librarian software systems. XML document containing bibliographic format data is input parameter. Accordingly, introduction of a new format for bibliographic material processing includes creating appropriate XML document which will be loaded into application as an input parameter. On the other hand, once bibliographic record is created, it is

placed into XML document which is output parameter of the editor. This XML document can be stored in various software systems for storing and searching data.

Поглавље 1

Увод

Резултати приказани у овој монографији добијени су у оквиру развоја библиотечког софтверског система БИСИС, и то оног дела који се односи на примену XML (Extensible Markup Language) технологија за развој едитора за обраду библиографске грађе. Основна улога овог едитора је креирање или модификација библиографских записа. Правила за креирање библиографских записа прописана су међународним стандардима.

Најпознатији стандард за размену библиографских података у машински читљивом облику је MARC [MARCFormat] (*Machine Readable Cataloging*) развијен пре више од 30 година у Конгресној библиотеци (*Library of Congress*) [LOC]. На основу овог формата, за потребе различитих држава и библиотечких система развили су се други библиографски формати, на пример: USMARC [USMARC] у Америци, CAN/MARC [CANMarc] у Канади, UKMARC [UKMARC] у Великој Британији, UNIMARC [UNIMARC94] (*Universal Machine Readable Cataloguing)* у европским државама. Године 1999. настао је формат MARC 21 [MARCStandards] на основу формата USMARC и CAN/MARC.

Од самог настанка XML спецификација уочена је повезаност структуре библиографских записа са структуром XML документа. Особина која повезују ове две структуре јесте постојање хијерхијског односа између елемената, елемената који могу бити обавезни или опциони, који се могу понављати или делити на поделементе. Свака XML структура може се формално описати XML шемом која је дефинисана XMLSchema језиком [XMLSchema]. Правила за креирање XML шеме одговарају правилима за креирање структуре MARC записа. Највећа предност коришћења XML-а за представљање библиографских података огледа се у једноставнијој обради података у отвореним дистрибуираним системима. Томе доприносе и већ развијени софтверски алати за рад са XML документима.

Постоји више пројеката који се баве применом XML технологија у развоју библиотечких софтверских система. Неки од њих су: MARC XML [MARCXML] у Француској, BiblioML [BiblioML], BookMARC [BookMARC] и БИСИС [BISIS04]. Пошто су резултати приказани у овој монографији добијену у развоју система БИСИС, на крају овог поглавља дат је кратак преглед развоја овог система.

1.1 Функционалност едитора за обраду библиографске грађе

Под едитором за обраду библиографске грађе подразумева се апликација која омогућава кориснику, библиотекару да уноси библиографске податке везане за одговарајућу публикацију онако како то прописује изабрани библиотечки стандард. Неке од основних функционалности које би овакав један едитор требало да обезбеди су следеће:

- ефикасан начин селекције елемената формата (поља, потпоља, индикатора,..) и унос података који припадају селектованим елементима
- једноставна измена унетих података
- учитавање записа ради модификације, или креирања новог записа на основу постојећег
- снимање записа
- селекцију шифара из унапред припремљених шифарника за шифрирана потпоља, индикаторе и позиције карактера ознаке слога и појединих контролних поља
- контрола унетих података (неке од тих контрола су провера поновљивости поља односно потпоља, провера формата података који се уносе, као и провера припадности унетих података дозвољеном скупу вредности, на пример да ли је шифра која је унета као вредност шифрираног потпоља или индикатора у шифарнику везаном за дато потпоље односно индикатор)
- могућност приказа података записа у различитим форматима, као што су разне врсте листића или лабела
- могућност увида у податке који се односе на сам формат (називи и особине поља, потпоља, и осталих елемената формата) ради адекватне едукације библиотекара који врши каталогизацију и смањења могућности грешке

У овом одељку биће дат преглед неколико постојећих апликација за обраду библиографске грађе. Неке од апликација приказаних у овом одељку доступне су путем Интернета, било као некомерцијални софтвери, било као демо верзије комерцијалних софтвера, док су подаци о другим апликацијама добављени путем упутства за употребу комерцијалних софтвара.

MarcEdit

MarcEdit [MarcEdit] је апликација написана у програмском језику *Perl* за оперативни систем *Windows,* њен аутор је Terry Reese са универзитета у Орегону (*Oregon State University*).

MarcEdit је апликација која у потпуности подржава функционалност за рад са библиогарфским записима. Састоји се од неколико делова, одвојених софтверских пакета, као што су Z39.50 клијент, апликације за конверзију података између различитих начина репрезентације записа и едитор за унос података о публикацијама.

Податке је у запис могуће додавати или модификовати директним уносом у текстуално поље у коме је приказан запис у пуном формату (*full format*). На овај начин омогућена је врло једноставна модификација, са једним недостатаком који се односи на одсуство било какве контроле над унетим подацима. Други начин уноса података је коришћење неких од функција понуђених у менију *Tools*.

Помоћни прозор за унос потпоља садржи падајући мени који омогућава селекцију поља, затим се у суседно тектуално поље уноси ознака потпоља, а затим у одговарајуће текстуално поље и текст. Оваквим уносом може доћи до грешке која се односи на невалидност записа. Узрок грешке могао би бити избор поља из падајућег менија који форматом није дефинисан, избор контролног поља (која не могу садржати потпоља) или уносом ознаке потпоља које није дефинисано за изабрано поље. За шифрирана потпоља не постоји начин селекције шифре из шифарника.

Polaris

Polaris је комерцијални софтверски производ америчке фирме *GIS Information Systems* [GISSystems], који у потпуности подржава каталогизацију по MARC21 стандарду.

Поред стандардних пречица за рад са текстом, основни прозор едитора садржи основне податке о запису, као што су контролни број, назив софтвера у ком је запис креиран, статус записа и наслов публикације на коју се запис односи. Највећи део основног прозора резервисан је за сам MARC 21 запис у *full* формату.

Селекцијом ознаке слога или контролног поља отвара се помоћни прозор у коме су понуђене шифре дозвољене за одговарајућу позицију карактера. Оваквим начином уноса вредности смањена је могућност грешке која се односи на унос погрешне шифре. Међутим, шифарници су обезбеђени само за ознаку слога и контролна поља, док код шифрираних потпоља не постоји овакав тип контроле.

IsisMarc

IsisMarc [ISIS] је у основи интерфејс за унос података у *CDS/ISIS* базе података и ради под оперативним системом *Windows*. Апликација представља унапређено окружење које је замена за стандардну *Winisis* екранску форму за унос података.

IsisMarc је *open source* производ *UNESCO*-овог сектора за комуникације и информације (*Communication and information Sector***)** у Француској, а настао је уз сарадњу ове институције са Конгресном библиотеком у Америци и SUI програма аргентинског министарства за образовање (*SIU Program of the Ministry of Education of Argentina*).

IsisMarc едитор омогућава кретање кроз колекцију записа и то или секвенцијалним кретањем кроз скуп записа или непосредном селекцијом записа према редном броју. Селекцијом одговарајућег записа омогућена је његова модификација, такође могуће је креирање новог записа који се додаје текућој колекцији.

Делови ознаке слога који су најрелевантнији за обраду записа издвојени су у горњем делу едитора, где је путем падајућег менија могуће изабрати код за одговарајућу позицију карактера у ознаци слога. Предност поменутог приступа састоји се у једноставном прегледу и измени информација битних за обраду записа, као што су статус записа, библиографски ниво, облик каталогизације и др. Могуће побољшање могло би се постићи навођењем текстуалног описа информације која се смешта у ознаку слога.

Испод дела за унос ознаке слога налазе се два падајућа менија за унос вредности индикатора. Ова два менија постају активна када се селектује поље за које је дефинисан индикатор и садрже могуће вредности индикатора за селектовано поље.

Свако поље у запису представљено је својом ознаком, описом информације која се чува у пољу и садржајем поља. Испред поља и потпоља која имају особину поновљивости налази се знак '%' чијом селекцијом је могуће додати ново поновљено поље, чиме је уведена контрола поновљивости. Селекцијом знака '±' испред поља које садржи потпоља отвара се листа потпоља дефинисана за селектовано поље. Знаком '?' испред поља представљена је пречица до спецификације поља у *html*-у. Ова спецификација одговара спецификацији са званичне Интернет адресе Конгресне библиотеке, али се датотеке које чине ову спецификацију јављају као саставни део инсталације софтвера. Оваквим приступом задовољена је особина једноставног увида у податке које се односе на сам формат.

Вредност контролних поља која садрже тачно дефинисане позиције карактера уносе се у посебном прозору. У горњем делу тог

прозора приказан је текући садржај поља подељен на позиције карактера, док централни део прозора заузима текстуални опис делова поља са могућношћу модификације вредности. У случају селекције позиције карактера која је шифрирана или уноса вредности шифрираног потпоља отвара се одговарајући шифарник из ког је могуће преузети код.

Двокликом миша на део за унос вредности нешифрираног потпоља отвара се проширено текстуално поље за унос. Оваквим решењем задовољена је особина едитора која се односи на могућност ефикасног уноса односно модификације унетих података. Уносом вредности у текстуално поље аутоматски се ажурира и одговарајућа вредност у линији која се односи на поље чије потпоље се модификује.

Concourse

Concourse [CONCOURSE] је библиотечки софтверски систем који је комерцијални производ фирме *Book Systems, Inc* [BOOKSYS] из САД-а. Као саставни део овог система понуђен је професионални едитор за унос библиографских података.

Модел едитора који ће бити представљен у овом одељку уводи специфичан приступ каталогитације по MARC 21. Специфичност се огледа у томе што се каталогизација може вршити на два начина, један начин не потразумева познавање стандарда, док се на други начин каталогизација врши у специјалном едитору, *Concourse Pro* за чије коришћење је потребно познавање стандарда.

Подаци о публикацији подељени су на логичке целине, на основу којих је унос подељен у неколико табова. У првом табу уносе се основни подаци о публикацији. Други таб носи назив *Analitics* и овде се уносе подаци који се односе на аналитику, као што су предметне одреднице, наслов серијске публикације, напомене и кратак садржај. Трећи таб, *Other* омогућава унос физичких карактеристика, као и остале наслове публикације.

Таб *Media* омогућава везивање мултимедијалних линкова за запис. Такође, овде је могуће унети URL адресу која се на било који начин односи на публикацију која се обрађује.

Таб који носи назив *MARC* пружа кориснику могућност прегледа записа у MARC 21 формату који је аутоматски изгенерисан на основу података унетих у прва четери таба. Овде није дозвољена модификација података.

Евентуално побољшање описаног решења састојало би се у томе да се уведе могућност дефинисања тагова од стране корисника са скупом поља за унос, због тога што се може јавити потреба за различитим груписањем података. Такође, одсуство података о формату могло би се

сврстати у недостатке описаног решења. При креирању едитора овог типа, као и приликом генерисања нових табова са пољима за унос посебна пажња мора се посветити пресликавању између поља за унос и елемената записа.

Описани едитор има могућност аутоматске провере правописа (*spell check*) и валидацију унетих података. Такође, уколико се обрађују записи који имају неке податке исте, могуће је једноставним пречицама са тастатуре копирати вредности претходно обрађеног записа.

1.2 Библиотечки софтверски систем БИСИС

Библиотечки софтверски систем БИСИС развија се од 1993. године. Тренутно је актуелна трећа верзија система. Систем је базиран на YUMARC формату [Vulić], који је настао из UNIMARC-а, COMARC-а и из захтева пројекта Библиотечка мрежа система научних и технолошких информација Србије. Развој и пројектовање овог система описан је у монографији [BISIS04].

Обрада библиографске грађе у актуелној верзији система БИСИС развијена је у Java окружењу и описана је у [Vidaković04].

У верзији 3 развијен је сопствени текст сервер за индексирање и претраживање библиографских записа у UNIMARC формату. Главне карактеристике овог сервера су: специјализованост која резултује бољим перформансама, трослојна архитектура, коришћење Java платформе и независност од коришћеног релационог система за управљање базом података. Подршка за Unicode стандард доследно је спроведена у целокупном систему БИСИС верзија 3.

Примена XML технологија у развоју библиотечког софтверског система БИСИС обухвата следеће:

- Опис библиграфских формата и библиографских записа помоћу XML шема језика
- Контролу квалитета XML библиографских записа
- Употребу XML native технологија
- Генерисање каталошких листића на основу XML библиографских записа
- Имплементацију XML едитора за библиографске формате и библиграфске записе

Опис библиграфских формата и библиографских записа помоћу XML шема језика разматран је са два аспекта. Први је формирање XML шеме тако да посебно описује све појединачне елементе формата а други

формирање XML шеме погодну за имплементацију библиотечког софтвера.

У радовима [Zeremski04, Zeremski01a, Zeremski01b] показано је да се помоћу XML шема језика могу моделирати сви концепти UNIMARC формата, као и сви елементи тих концепата. Овако формирана XML шеме могла би се користити за валидацији изабраног дела UNIMARC формата за обраду библиографске грађе. Такође је показано да се помоћу XML шема језика могу моделирати сви концепти UNIMARC библиографских записа, као и сви елементи тих концепата. XML шема бибиографских записа описана је у радовима [Zeremski01c, Zeremski02, Zeremski01d]. Овако формирана XML шема могла би се користити за валидацију библиографских записа формираних по UNIMARC формату као и за контролу квалтета библиографских записа.

Међутим, у радовима [Dimić07a, Milosavljević07] дат је предлог XML шема које описују концепте библиографских формата (YUMARC, UNIMARC, MARC21) као што су поља, потпоља, индикатори, шифарници, док појаве тих концепата садрже конкретне податке о формату, односно спецификацију појединачних поља, потпоља, индикатора и друго. Поред тога, у овим радовима описане су и XML шеме библиографских записа формираних по тим форматима. Овакве шеме омогућавају да се у софтверским системима ради само са изабраним деловима формата за обраду библиографске грађе.

Контрола квалитета XML библиографских записа. Монографија [Budimir04] посвећена је контроли квалитета библиографских записа. Дефинисана је и систематизована контрола за утврђивање квалитета библиографских записа формираних по UNIMARC формату. Концепти контрола записа описани су помоћу XML шема језика. Контроле које се не могу описати XML шема језиком описане су помоћу XSLT спецификација. Имплементиран је систем за контролу квалитета XML библиографских записа у Java окружењу.

Употреба XML native технологија. Циљ истраживања приказаних у радовима [Škrbić04, Škrbić05] је испитивање могућности употребе XML native технологија у развоју библиотечког информационог система заснованог на UNIMARC формату. У складу са тим, урађено је моделирање и имплементација софтверског система за анализу и верификацију употребе XML native технологија за обраду библиографске грађе.

Генерисање каталошких листића на основу XML библиографских записа. У радовима [Rađenović06a, Rađenović06b] приказано је моделирање и имплементација софтверског пакета за формирање библиотечких каталошких листића. Формирање ових листића базирано је на софтверском пакету FreeMarker и XML документима библиографских

записа. Архитектура система је таква да омогућује доступност каталошких листића из свих сегмената библиотечког софтверског система БИСИС и могућност ажурирања каталошких листића без поновног компајлирања изворног кода.

Имплементацију XML едитора за библиографске формате и библиграфске записе. Детаљан опис функционалности едитора за обраду библиографске грађе у актуелној верзији система БИСИС дато је у [BISIS03]. Поред едитора за обраду библиографске грађе у оквиру развоја система БИСИС вршено је и истраживање које се односи на моделирање и имплементацију едитора за UNIMARC формат. У тези [Mijić03] дат је предлог моделирања и имплементације XML едитора за опис UNIMARC формата. У монографији [Škrbić05] приказано је моделирање и имплементација XML едитора који илуструје начин употребе XML native технологија у развоју библиотечког информационог система заснованог на UNIMARC стандарду. Као наставак ових истраживања, у овој монографији приказано је моделирање и имплементација XML едитора за обраду библиогарфске грађе по YUMARC формату који је интегрисан у нову четврту верзију система БИСИС.

Поглавље 2

XML шеме YUMARC формата и библиографског записа по YUMARC формату

Од самог настанка XML технологије појавило се интересовање за њеном применом у библиотечким информационим системима. Увођење XML-а у библиотечке системе до сада је углавном подразумевало мапирање библиографских записа на XML документе. Сврха представљања библиографских записа у XML-у најчешће је била лакши начин њиховог преноса.

На основу UNIMARC формата развијени су и национални библиографски формати као што је YUMARC формат који се користи у библиотечком софтверском систему БИСИС.

XML технологије се у оквиру новог едитора за обраду библиографске грађе користе за опис библиографских записа али и података који се односе на сам библиотечки формат или подскуп формата по коме се врши обрада. XML шема библиографског записа даје модел XML документа за складиштење записа док XML шема библиотечког формата представља модел за XML документ у ком ће се складиштити подаци о елементима самог формата. У податке о елементима формата спадају називи елемената формата, њихов опис и особине, као и разна ограничења која се односе на правила за обраду библиографске грађе.

За оба ова случаја коришћења XML-а потребно је извршити моделирање у XML технологији коришћењем XMLSchema језика [XMLSchema] да би се XML документи могли користити у едитору.

У раду [Milosavljević07] дат је предлог XML шеме YUMARC формата, као и XML шеме библиографских записа по овом формату. Едитор који је тема ове монографија подржава рад са XML документима који се инстанце ових шема и оне ће бити детаљно описане у овом поглављу. Поред тога, биће дати и примери XML докумената који представљају инстанце ових шема.

2.1 YUMARC формат

Према YUMARC формату, библиографски запис са састоји од низа *поља*. Поље се састоји од низа *потпоља*, а могу му бити додељена и највише два *индикатора*. Потпоља представљају основне носиоце података о публикацији, док је улога индикатора да ближе одреде значење података у пољу или појаву податка у испису.

Поља су дефинисана ознаком која се састоји од три цифре и подељени су у десет блокова. Прва цифра у ознаци поља указује на блок коме поље припада (на пример поље са ознаком 200 припада блоку 2 чији је назив Блок главног описа). Према YUMARC стандарду поље карактеришу особине поновљивост, обавезност и секундарност. Поновљивост поља значи да се поље може појавити више пута у запису а обавезност да се поље мора појавити у запису. Поља која имају особину секундарности могу се појавити као садржај потпоља поља из блока 4 (Блок за повезивање библиографских записа).

Подаци у пољима подељени су у више потпоља. Потпоља су, у оквиру поља дефинисана својом ознаком која је један алфанумерички знак и могу имати особину поновљивости у оквиру поља и особину обавезности.

Податак у потпољу може бити текст или шифра из неког од шифарника YUMARC формата. За нека потпоља може бити дефинисана максимална дозвољена дужина податка. Такође, постоје потпоља која имају дефинисане подразумеване вредности (*default value*), односно вредности које се унапред предвиђене као садржај потпоља. Шифарници дефинисани YUMARC форматом могу важити у оквиру једног потпоља и тада се ради о *интерним шифарницима*, или могу важити за више потпоља што је карактеристика *екстерних шифарника*. Екстерни шифарници су разврстани по типовима, као на пример, шифарник земаља, језика, кодова ауторства, итд.

Блок 4 је блок за повезивање библиографских записа. Веза између записа остварује се уносом текстуалних или нумеричких података у одговарајућа поља или потпоља блока 4. Ти подаци могу бити идентификациони број записа са којим се врши повезивање или нова, секундарна поља којима се описује документ са којим вршимо повезивање.

Блок 9 дефинише се у националном оквиру. Намењен је за унос података који су од локалног значаја за библиотечку мрежу или појединачну библиотеку. У овом блоку постоје потпоља чији су подаци структуирани у још један ниво подструктуре - *потпотпоља*. Потпотпоља имају исте особине као и потпоља, а постоје и шифарници за потпотпоља.

Помоћу YUMARC формата могуће је, користећи различите скупове поља и потпоља, описати све типове публикације: књиге, часописе, новине, докторске дисертације, чланке, музичка дела, картографске публикације, рукописе, рачунарске датотеке, итд.

Следећи пример илуструје обраду једне монографске публикације по YUMARC стандарду. Нека је дата публикација са насловом "Concepts of Programming Languages". Аутор ове књиге је Robert W. Sebesta, а међународни стандардни број за књигу (ISBN број) је 0-8053-7133-8. Језик публикације је енглески а земља издавања САД. Што се локацијских података тиче, у библиотеци постоји један примерак ове публикације са сигнатуром М-20111, и инвентарним бројем 000020111. На листингу 1.1 приказан је YUMARC запис за наведену публикацију.

```
001 ## [7]ba[a]i[b]a[c]m[d]0
010 ## [a]0-8053-7133-8
100 ## [b]d[c]1996[h]scr
101 0# [a]eng
102 ## [a]usa
105 ## [a]a
200 0# [a]Concepts of Programming Languages[f]Robert W.
Sebesta
205 ## [a]3. izd.
210 ## [a]Reading [etc.][c]Addison-Wesley Publishing
Company[d]1996
215 ## [a]xv, 634 str.[c]ilustr.[d]24 cm
700 #1 [4]070[a]Sebesta[b]Robert W.
992 ## [b]crsale9701-old
996 0# [d]<l>M<n>20111[f]000020111
```

Листинг 1.1 *Пример YUMARC записа*

Запис је приказан у више редова, по један ред за свако поље записа. Прва три реда у запису представљају идентификатор поља, следећа два представљају вредности првог и другог индикатора (уколико им вредност није дефинисана наводи се знак #). Ознаке потпоља уписане су између знакова [и], а иза њих наводи се садржај потпоља. Ознаке потпотпоља уписане су између знакова < и >.

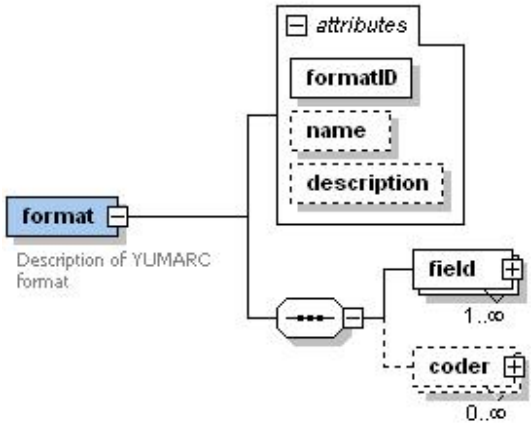
У наставку ће бити дат опис структуре библиографског записа приказаног на листингу 1.1. Блок 0 је блок за идентификацију. У поље *001* уписују се неопходни кодирани подаци који опсисују публикацију. Јединствени ISBN број књиге је садржај поља *010* потпоље *a*. Поља из блока 1 предвиђена су за кодиране информације о типу публикације, језику каталогизације, језику публикације и слично. Кодови се преузимају из неког од шифарника YUMARC формата. Главни стварни наслов књиге смештен је пољу *200*, потпоље *a*, а аутор у потпољу *f*. Подаци о аутору уносе се и у поље *700* при чему други индикатор овог поља одрађује да ли

се у потпољу *a* уноси име или презиме аутора. Поље *210* предвиђено је за податке о издању, а *215* за материјелни опис публикације. У потпоља поља *996* уносе се подаци о сигнатури и инвентарном броју појединачног примерка публикације. Потпоље *d* садржи потпотпоља за структуирање података из сигнатуре. Тако се у потпотпоље *l* уноси податак о локацији на којој је примерак књиге смештен, док се у потпотпоље *n* уноси број који одређује место на наведеној локацији.

2.2 XML шема YUMARC формата

Модел спецификације YUMARC формата представљен је XML шемом чија је инстанца документ у ком ће се складиштити подаци о елементима формата.

Графички приказ XML шеме YUMARC формата дат је на слици 2.1. Коренски елемент је *format* и он има три атрибута *formatID*, *name* и *description*, који предтсвљају редом идентификациони број формата, назив и опис формата. Елемент *format* садржи низ поделемената који се односе на спецификацију поља формата и низ елемената који се односе на екстерне шифарнике.



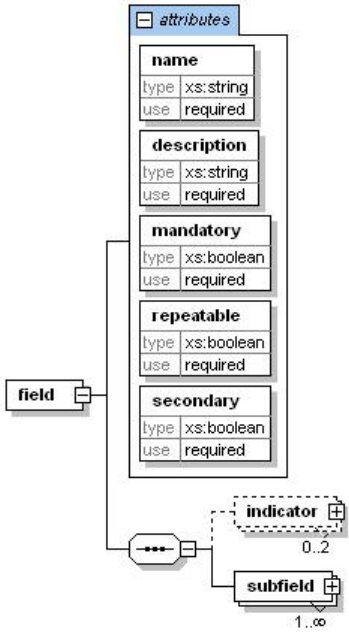
Слика 2.1 *Коренски елемент XML шеме YUMARC формата*

Спецификација поља формата моделирана је елементом *field*, чији детаљан приказ је дат на слици 2.2. Поље је у XML шеми моделирано тако да садржи пет атрибута који га ближе одређују, највише два поделемента којима се дефинишу индикатори и низ поделемената којима се специфицирају потпоља посматраног поља.

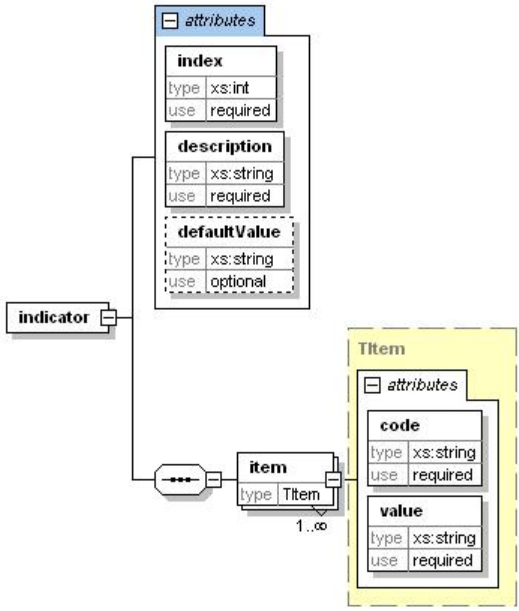
Елемент *field* садржи следеће атрибуте:

- *name* - троцифрана ознака поља,
- *description* - опис поља, односно опис библиотечких података које према правилима YUMARC формата поље садржи,
- *mandatory* - логичка вредност којом се дефинише да ли је поље обавезно,
- *repeatable* - логичка вредност која дефинише особину поновљивости,
- *secondary* - логичка вредност која одређује да ли је поље секундарно.

Елементом *indicator* моделирана је спецификација индикатора. Кардиналитет појављивања овог елемента у оквиру поља на које се односи је *0..2* чиме је моделиран услов да поље може имати највише два индикатора, али не мора имати ни један. Детаљна спецификација елемента *indicator* приказана је на слици 2.3.



Слика 2.2 *Модел поља YUMARC формата – елемент field*



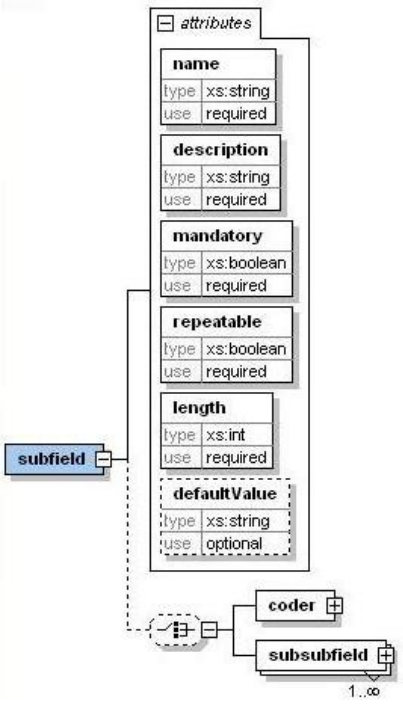
Слика 2.3 *Модел индикатора – елемент indicator*

Индикатор одређују три атрибута елемента *indicator*, а то су:

- *index* – може имати вредност *1* или *2* у зависности да ли се ради о првом или другом индикатору поља
- *description* – опис индикатора, односно опис информације коју индикатор носи према правилима дефинисаним YUMARC стандардом
- *defaultValue* – подразумевана вредност индикатора, односно вредност коју ће индикатор у библиографском запису имати у случају да му се не додели друга вредност, употреба овог атрибута је опционална

Елемент *indicator* је секвенца елемената *item* којим је моделирана могућа вредност за индикатор и типа је *Titem*, који је дефинисан глобално. Тип *TItem* дефинише два атрибута, атрибут *code* који представља вредност коју индикатор може имати у библиографском запису, и атрибут *value* који представља текстуални опис значења те вредности.

Детаљан приказ елемента *subfield* дат је на слици 2.4. Спецификацију потполја YUMARC стандарда одређује шест атрибута, а као поделемент елемента који представља потполје формата може се јавити шифарник потполја, моделиран елементом *coder* или низ елемената који представљају потпотполја дефинисана по YUMARC стандарду.



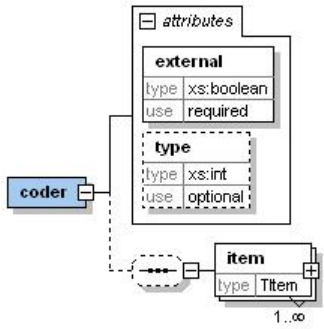
Слика 2.4 *Модел спецификације потпоља – елемент subfield*

Атрибути елемента *subfield* имају следећа значења:

- *name* – ознака потпоља која је један алфанумерички знак
- *description* - опис потпоља, односно опис библиотечких података које према правилима YUMARC формата потпоље садржи
- *mandatory* - логичка вредност којом се дефинише да ли је потпоље обавезно
- *repeatable* - логичка вредност која дефинише особину поновљивости потпоља у оквиру поља
- *length* – дозвољена дужина податка у потпољу
- *defaultValue* – подразумевана вредност потпоља, односно вредност коју ће потпоље у библиографском запису имати у случају да му се не додели друга вредност, употреба овог атрибута је опционална

Шифарник потпоља моделиран је елементом *coder* чије детаљан приказ је дат на слици 2.5. Атрибут *external* је логичког типа и он има вредност *true* уколико потпоље на које се шифарник односи узима вредност из неког екстерног шифарника и тада атрибут *type* има нумеричку вредност која је додељена одговарајућем екстерном

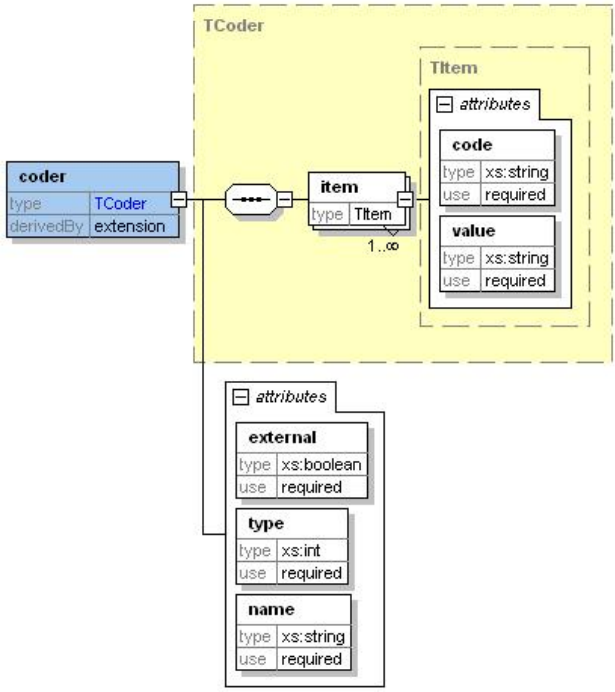
шифарнику. У посматраном случају неће се јавити секвенца елемената *item* чија употреба је опционална. У случају да је за потпоље везан интерни шифарник, атрибут *external* ће имати вредност *false*, а атрибут *type* неће имати дефинисану вредност. Овакав случај је предвиђен шемом јер је употреба атрибута *type* опционална. У случају интерног шифарника за потпоље, у оквиру елемента *coder* јавља се секвенца поделемената *item* глобално дефинисаног типа *TItem*.



Слика 2.5 *Модел шифарника потпоља – елемент coder*

Спецификација потпотпоља YUMARC формата подудара се са спецификацијом потпоља, једина разлика у моделу ова два елемента састоји се у томе да потпотпоља као поделемент могу имати само елемент који се односи на шифарник потпотпоља и не могу имати још један ниво подструктуре.

На слици 2.6 дат је графички приказ модела екстерног шифарника који се јавља као поделемент коренског елемента *format*. Екстерни шифарник моделиран је елементом *coder* који представља проширење типа *TCoder* атрибутима *external*, *type* и *name*. Тип *TCoder* је дефинисан као секвенца ставки шифарника представљених елементом *item* који је раније описан у раду. Атрибут *external* за екстерни шифарник добија вредност *true*. Атрибут *type* представља јединствену нумеричку вредност за екстерни шифарник ради референцирања из потпоља које ће из њега узимати вредност. Атрибут *name* моделира текстуални опис екстерног шифарника.



Слика 2.6 *Модел екстерног шифарника*

Листинг XML шеме YUMARC формата. У наставку је дат листинг комплетне XML шеме YUMARC формата.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://bisis.ns.ac.yu/schemas/unimarc.xsd"
xmlns="http://bisis.ns.ac.yu/schemas/unimarc.xsd" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="format">
  <xs:annotation>
          <xs:documentation>Opis konfiguracije UNIMARC formata</xs:documentation>
  </xs:annotation>
  <xs:complexType>
  <xs:sequence>
    <xs:element name="field" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="indicator" minOccurs="0" maxOccurs="2">
                                                    <xs:complexType>
                                                     <xs:sequence>
                                                       <xs:element name="item"
type="TItem" maxOccurs="unbounded"/>
```

```
                                    </xs:sequence>
                                     <xs:attribute name="index" type="xs:int" use="required"/>
                                                                                    <xs:attribute
name="description" type="xs:string" use="required"/>
                                                                <xs:attribute name="defaultValue"
type="xs:string" use="optional"/>
                              </xs:complexType>
    </xs:element>
    <xs:element name="subfield" maxOccurs="unbounded">
      <xs:complexType>
      <xs:choice minOccurs="0">
       <xs:element name="coder">
          <xs:complexType>
            <xs:sequence minOccurs="0">
               <xs:element name="item" type="TItem" maxOccurs="unbounded"/>
             </xs:sequence><
             <xs:attribute name="external" type="xs:boolean" use="required"/>
              <xs:attribute name="type" type="xs:int" use="optional"/>
           </xs:complexType>
       </xs:element>
        <xs:element name="subsubfield" maxOccurs="unbounded">
           <xs:complexType>
            <xs:sequence minOccurs="0">
               <xs:element name="coder">
                  <xs:complexType>
                    <xs:sequence minOccurs="0">
                       <xs:element name="item" type="TItem" maxOccurs="unbounded"/>
                    </xs:sequence>
                     <xs:attribute name="external" type="xs:boolean" use="required"/>
                     <xs:attribute name="type" type="xs:int" use="optional"/>
                  </xs:complexType>
                </xs:element>
             </xs:sequence>
             <xs:attribute name="name" type="xs:string" use="required"/>
             <xs:attribute name="description" type="xs:string" use="required"/>
             <xs:attribute name="mandatory" type="xs:boolean" use="required"/>
             <xs:attribute name="repeatable" type="xs:boolean" use="required"/>
             <xs:attribute name="length" type="xs:int" use="required"/>
             <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
          </xs:complexType>
         </xs:element>
      </xs:choice>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="description" type="xs:string" use="required"/>
      <xs:attribute name="mandatory" type="xs:boolean" use="required"/>
        <xs:attribute name="repeatable" type="xs:boolean" use="required"/>
        <xs:attribute name="length" type="xs:int" use="required"/>
        <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
      </xs:complexType>
```

```
    </xs:element>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="description" type="xs:string" use="required"/>
   <xs:attribute name="mandatory" type="xs:boolean" use="required"/>
   <xs:attribute name="repeatable" type="xs:boolean" use="required"/>
   <xs:attribute name="secondary" type="xs:boolean" use="required"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="coder" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:complexContent>
        <xs:extension base="TCoder">
        <xs:attribute name="external" type="xs:boolean" use="required"/>
        <xs:attribute name="type" type="xs:int" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
       </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="formatID" type="xs:integer" use="required"/>
 <xs:attribute name="name" type="xs:string" use="optional"/>
 <xs:attribute name="description" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
<xs:complexType name="TCoder">
 <xs:sequence>
  <xs:element name="item" type="TItem" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
 <xs:complexType name="TItem">
  <xs:attribute name="code" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string" use="required"/>
 </xs:complexType>
</xs:schema>
```

2.3 XML документ YUMARC формата

XML документ који представља појаву XML шеме приказану у претходном одељку садржи спецификацију YUMARC формата.

На листингу 2.2 приказан је део XML документа спецификације YUMARC формата који се односи на спецификацију поља *200*. Атрибут *name* садржи назив поља, док атрибут *description* садржи текст *"Stvarni naslov i podaci o odgovornosti"* који представља опис поља *200*. Атрибут *mandatory* има вредност *true* што значи да је поље обавезно. Поље *200* има

дефинисан први индикатор који је у посматраном XML сегменту представљен угњежденим елементом *indicator* са вредношћу атрибута *index* једнаком *1*. Опис индикатора је вредност атрибута *description* елемента *indicator*. Подразумевана вредност за посматрани индикатор је *0* и препознаје се из вредности атрибута *defaultValue* елемента *indicator*.

```
<field name="200" description="Stvarni naslov i podaci o odgovornosti" mandatory="true"
repeatable="false" secondary="true">
        <indicator index="1" description="Indikator važnosti glavnog stvarnog naslova"
defaultValue="0">
                <item code="0" value="Naslov nije značajan"/>
                <item code="1" value="Naslov je značajan"/>
        </indicator>
        <subfield name="a" description="Glavni stvarni naslov" mandatory="true"
repeatable="false" length="0"/>
        <subfield name="b" description="Opšta oznaka građe" mandatory="false"
repeatable="false" length="0"/>
        <subfield name="c" description="Glavni stvarni naslov drugog autora"
mandatory="false" repeatable="false" length="0"/>
        <subfield name="d" description="Uporedni stvarni naslov" mandatory="false"
repeatable="false" length="0"/>
        <subfield name="e" description="Podnaslov" mandatory="false" repeatable="false"
length="0"/>
        <subfield name="f" description="Prvi podatak o odgovornosti" mandatory="false"
repeatable="false" length="0"/>
        <subfield name="z" description="Jezik uporednog stravrnog naslova"
mandatory="false" repeatable="false" length="3">
                <coder external="true" type="1"/>
        </subfield>
</field>
```

Листинг 2.2 *XML дукумент YUMARC формата*

Као поделементи елемента којим се специфицира поље *200* јављају се елементи *subfield* који садрже спецификацију потпоља поља *200* по YUMARC стандарду. Потпоље *z* поља *200* које је описано текстом *"Jezik uporednog stravrnog naslova"*, није обавезно, нити поновљиво и дозвољена дужина му је *3*. Ово потпоље је шифрирано, што је у XML документу представљено угњежденим елементом *coder*. Вредност атрибута *external* елемента *coder* је *true* што значи да потпоље узима вредности из екстерног шифарника. Идентификација екстерног шифарника за потпоље остварује се на основу вредности атрибута *type* елемента *coder*.

На листингу 2.3 приказан је део XML документа спецификације формата који се односи на спецификацију потпоља *a* поља *105* које узима вредности из интерног шифарника. Интерни шифарник је у посматраном примеру представљен елементом *coder* који је поделемент елемента

subfield за спецификацију потпоља *a*. Како се ради о интерном шифарнику, атрибут *external* има вредност *false.* Као поделементи елемента *coder* јављају се појаве елемената *item* чијим атрибутима је описана ставка интерног шифарника. Шифра *d* има опис *"grafički prikazi"* који се јавља као вредност атрибута *value* одговарајућег елемента *item*.

```
<field name="105" description="Polje kodiranih podataka: knjige" mandatory="false"
repeatable="false" secondary="false">
        <subfield name="a" description="Kodovi za ilustracije" mandatory="false"
repeatable="false" length="1">
                <coder external="false">
                        <item code="a" value="ilustracije"/>
                        <item code="b" value="zemljopisne karte"/>
                        <item code="c" value="portreti"/>
                        <item code="d" value="grafički prikazi"/>
                        <item code="e" value="planovi"/>
                        <item code="f" value="table (listovi, koji sadrže slikovnu građu...)"/>
                        <item code="g" value="note"/>
                        <item code="h" value="faksimili"/>
                        <item code="i" value="grbovi"/>
                        <item code="j" value="genealoške table"/>
```

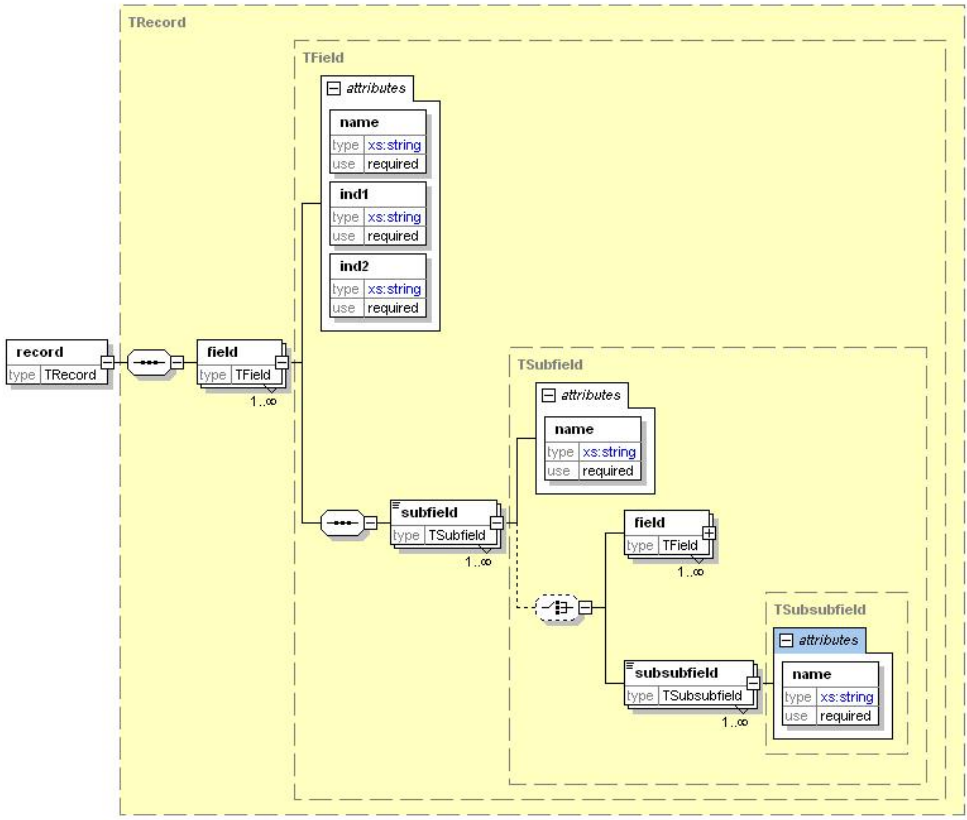
Листинг 2.3 *XML документ YUMARC формата*

2.3 XML шема библиографског записа по YUMARC формату

XML шема библиографског записа представља модел репрезентације записа у форми XML документа [Milosavljević07]. Графички приказ XML шеме дат је на слици 2.7.

Коренски елемент XML документа којим је представљен библиографски запис је елемент *record* и он је сложеног типа *TRecord*. Тип *TRecord* је дефинисан као секвенца елемената *field* коим је моделирано поље записа. Елемент *field* је сложеног типа *TField* има дефинисана три атрибута и садржи секвенцу елемената *subfield*. Атрибут *name* елмента *filed* моделира назив поља, док атрибути *ind1* и *ind2* одговарају вредностима првог и другог индикатора поља.

Елементом *subfield* представљена су потпоља записа и овај елемент је сложеног типа *TSubfield*. Назив потпоља смешта се у атрибут *name*, а потпоље може као поделементе садржати друга поља или потпотпоља. Поља која се јављају као садржај потпоља су секундарна поља, али имају исте особине као и обична поља у запису, због чега су и ова поља типа *TField*. Елементом *subsubfield* моделирано је потпотпоље записа. Назив потпотпоља јавиће се као вредност атрибута *name* елемента *subsubfield*.



Слика 2.7 *Графичка репрезентација XML шеме библиографског записа по YUMARC формату*

Листинг XML шеме библиографских записа по YUMARC стандарду. У наставку је дат листинг комплетне XML шеме библиографских записа по YUMARC формату.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="setofrecords">
  <xs:complexType>
            <xs:sequence>
              <xs:element name="record" type="TRecord" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<xs:element name="record" type="TRecord"/>
  <xs:complexType name="TRecord">
            <xs:sequence>
              <xs:element name="field" type="TField" maxOccurs="unbounded"/>
             </xs:sequence>
            </xs:complexType>
            <xs:complexType name="TField">
              <xs:sequence>
                      <xs:element name="subfield" type="TSubfield" maxOccurs="unbounded"/>
               </xs:sequence>
               <xs:attribute name="name" use="required">
                        <xs:simpleType>
                          <xs:restriction base="xs:string">
                                  <xs:length value="3"/>
                                  <xs:pattern value="\d{3,3}"/>
                          </xs:restriction>
                        </xs:simpleType>
                       </xs:attribute>
                       <xs:attribute name="ind1" use="required">
                         <xs:simpleType>
                           <xs:restriction base="xs:string">
                                    <xs:length value="1"/>
                           </xs:restriction>
                         </xs:simpleType>
                       </xs:attribute>
                       <xs:attribute name="ind2" use="required">
                         <xs:simpleType>
                                  <xs:restriction base="xs:string">
                              <xs:length value="1"/>
                                  </xs:restriction>
                         </xs:simpleType>
                        </xs:attribute>
              </xs:complexType>
              <xs:complexType name="TSubfield" mixed="true">
                      <xs:choice minOccurs="0">
                        <xs:element name="field" type="TField" maxOccurs="unbounded"/>
                        <xs:element name="subsubfield" type="TSubsubfield"
maxOccurs="unbounded"/>
                      </xs:choice>
                      <xs:attribute name="name" use="required">
                        <xs:simpleType>
                          <xs:restriction base="xs:string">
                           <xs:length value="1"/>
                          </xs:restriction>
                        </xs:simpleType>
                      </xs:attribute>
              </xs:complexType>
              <xs:complexType name="TSubsubfield">
                      <xs:simpleContent>
```

```
                    <xs:extension base="xs:string">
                            <xs:attribute name="name" use="required">
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:length value="1"/>
                                </xs:restriction>
                            </xs:simpleType>
                           </xs:attribute>
                     </xs:extension>
                   </xs:simpleContent>
            </xs:complexType>
</xs:schema>
```

2.4 XML документ библиографског записа по YUMARC формату

На листингу 2.4 приказан је XML документ библиографског записа. У оквиру почетног и завршног тага елемента *record* налази се XML репрезентација библиографског записа по YUMARC стандарду.

Поља су у запису сортирана према ознаци. Друго поље у запису је поље *010* његова ознака јавља се као садржај атрибута *name* другог по реду елемента *field*. Ово поље нема дефинисане индикаторе због чега је вредност атрибута *ind1* и *ind2* бланко знак. Поље *010* у посматраном библиографском запису садржи потпоље *a* са вредношћу *0-8053-7133-8*. Наведено потпоље је представљено елементом *subfield* који се јавља као поделемент елмента којим је представљено поље *010*. Садржај потпоља наведен је између почетног и завршног тага елемента *subfield* чији је атрибут *name* једнак *a*.

Вредност првог индикатора поља *200* је *0*, што је у XML документу библиографског записа представљено атрибутом *ind1="0"* елемента *field* са вредношћу атрибута *name* једнакој *200*.

Потпоље *d* поља *996* садржи потпотпоља која су представљена елементима *subsubfield* у оквиру почетног и завршног тага одговарајућег елемента *subfield*. Прво потпотпоље има назив *l* који се може прочитати из вредности атрибута *name*. Садржај овог потпотпоља је *M* и он је наведен између почетног и завршног тага одговарајућег елемента *subsubfield*.

```
<record>
  <field name="001" ind1=" " ind2=" ">
    <subfield name="7">ba</subfield>
    <subfield name="a">i</subfield>
    <subfield name="b">a</subfield>
    <subfield name="c">m</subfield>
    <subfield name="d">0</subfield>
  </field>
  <field name="010" ind1=" " ind2=" ">
    <subfield name="a">0-8053-7133-8</subfield>
  </field>
  <field name="100" ind1=" " ind2=" ">
    <subfield name="b">d</subfield>
<subfield name="c">1996</subfield>
    <subfield name="h">scr</subfield>
  </field>
  <field name="101" ind1="0" ind2=" ">
    <subfield name="a">eng</subfield>
  </field>
  <field name="102" ind1=" " ind2=" ">
    <subfield name="a">usa</subfield>
  </field>
  <field name="105" ind1=" " ind2=" ">
    <subfield name="a">a</subfield>
  </field>
  <field name="200" ind1="0" ind2=" ">
    <subfield name="a">Concepts of Programming Languages</subfield>
    <subfield name="f">Robert W. Sebesta</subfield>
  </field>
  <field name="205" ind1=" " ind2=" ">
    <subfield name="a">3. izd.</subfield>
  </field>
  <field name="210" ind1=" " ind2=" ">
    <subfield name="a">Reading [etc.]</subfield>
    <subfield name="c">Addison-Wesley Publishing Company</subfield>
    <subfield name="d">1996</subfield>
  </field>
  <field name="215" ind1=" " ind2=" ">
    <subfield name="a">xv, 634 str.</subfield>
    <subfield name="c">ilustr.</subfield>
    <subfield name="d">24 cm</subfield>
  </field>
  <field name="700" ind1=" " ind2="1">
    <subfield name="4">070</subfield>
    <subfield name="a">Sebesta</subfield>
    <subfield name="b">Robert W.</subfield>
  </field>
  <field name="992" ind1=" " ind2=" ">
    <subfield name="b">crsale9701-old</subfield>
```

```
  </field>
  <field name="996" ind1="0" ind2=" ">
    <subfield name="d">
      <subsubfield name="l">M</subsubfield>
      <subsubfield name="n">20111</subsubfield>
    </subfield>
    <subfield name="f">000020111</subfield>
  </field>
</record>
```

Листинг 2.4 *XML документ библиографског записа*

Поглавље 3

Моделирање XML едитора за обраду библиографске грађе

У овом поглављу дат је опис информационих захтева XML едитора за обраду библиографске грађе путем UML нотације. Функционалност система описана је путем дијаграма случајева коришћења. Статички модел система дат је путем дијаграма пакета, а сваки од пакета описан је посебним дијаграмом класа. Динамички модел едитора дат је путем дијаграма активности.

3.1 Дијаграм случајева коришћења

Целокупна функционалност едитора за обраду библиографске грађе може се поделити у неколико случајева коришћења. Дијаграм случајева коришћења приказан је на слици 3.1.

У наставку ће бити дат текстуални опис случајева коришћења који описују функционалност система за обраду библиограгске грађе.

Случај коришћења: Подешавање окружења

Кратак опис: Избор типа обраде

Учесници: Библиотекар

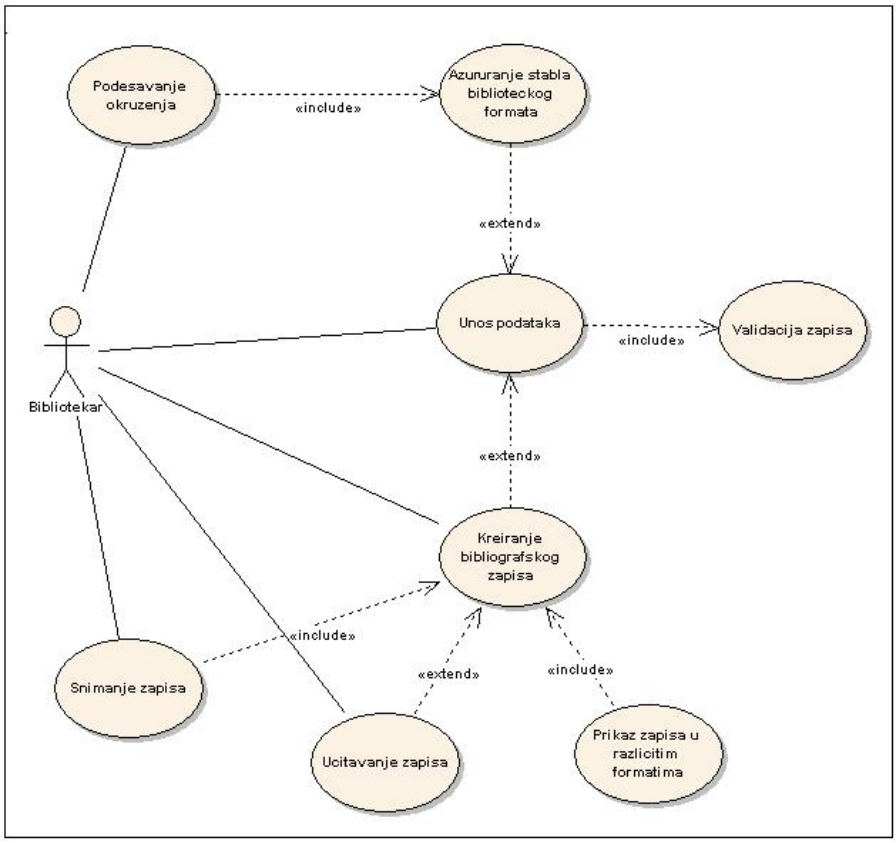
Услови који морају бити задовољени пре извршавања: Постоји дефинисан скуп типова обраде за пријављеног библиотекара

Опис: За сваког библиотекара у систему дефинисан је скуп типова обраде по којима он може да врши каталогизацију. Један тип обраде дефинисан је типом публикације на који се односи, иницијалним скупом потпоља и скупом обавезних потпоља. За сваког библиотекара дефинише се и подрезумевајући (*default*) тип обраде на основу кога се врши ажурирање стабла библиотечког формата приликом првог покретања едитора. У овај случај коришћења спада и функционалност која се односи на промену типа обраде при чему се стабло библиотечког формата ажурира на основу изабраног типа обраде. Библиотекар може изабрати само онај тип обраде који је за њега дефинисан. Ажурирање стабла библиотечког формата описано је посебним случајем коришћења. Над ажурираним стаблом

формата могуће је вршити селекцију делова формата ради уноса вредности.

Изузеци: нема

Услови који морају бити задовољени после извршавања: Издвојен је подскуп формата на основу изабраних параметара и ажурирано је стабло библиотечког формата



Слика 3.1 *Дијаграм случајева коришћења едитора за обраду библиографске грађе*

Случај коришћења: Ажурирање стабла библиотечког формата

Кратак опис: Скуп операција које је могуће извршити над подскупом формата приказаном у облику стабла

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Постоји потреба за променом подскупа формата или селекцијом дела формата

Опис: Врши се промена подскупа формата представљеног у облику стабла. Над стаблом библиотечког формата могуће је вршити додавање или уклањање елемената (поља или потпоља) или селекција елемената за унос података о запису.

Изузеци: нема

Услови који морају бити задовољени после извршавања: Стабло формата садржи све неопходне елементе, не садржи непотребне елементе или је селектован жељени део формата

Случај коришћења: Унос података

Кратак опис: Уносе се подаци о публикацији у складу са правилима стандарда

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Постоји податак о публикацији који треба унети у запис

Опис: Уноси се податак о публикацији која се обрађује. На почетку уноса врши се селекције елемента формата за који се уноси податак. Уколико се елемент за који се уноси податак не налази у стаблу долази до реализације случаја коришћења *Ажурирање стабла библиотечког формата* и елемент са прво додаје у стабло па затим селектује. У процес уноса података укључена је и валидација унете вредности. Уколико је податак валидан реализује се проширење случај коришћења случајем коришћења *Креирање библиографског записа* којим се податак додаје у запис. У случају да податак није валидан прослеђује се одговарајућа порука.

Изузеци: нема

Услови који морају бити задовољени после извршавања: Податак је учитан у меморију и спреман да се дода у запис или је послата порука о грешци у случају невалидности унетих података

Случај коришћења: Креирање библиографског записа

Кратак опис: Додавање нових или измена података у запису

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Податак о

запису је унет и извршена је валидација унетог податка

Опис: На основу података о публикацији креира се библиографски запис по усвојеном стандарду. Под креирањем записа подразумева се процес додавања елемената (поља, потпоља, индикатора) и модификација већ унетих вредности. У случају да је потребно изменити постојећи запис или се нови запис креира изменом вредности постојећег записа случај коришћења се проширује случајем коришћења *Учитавање записа*

Изузеци: нема

Услови који морају бити задовољени после извршавања: Над записом је извршена измена која се односи на додавање новог или измену већ унетог елемента. Тако модификован запис приказује се на екранској форми едитора

Случај коришћења: Валидација записа

Кратак опис: Врши се контрола исправности библиографског записа и његова усклађеност са изабраним стандардом

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Постоји подаци о формату на основу којих се врши валидација и постоји библиографски запис који се валидира

Опис: Врши се провера исправности података библиографског записа. Постоји неколико врста провера исправности, а неке од њих су да ли запис задовољава правила стандарда, да ли су унети подаци одговарајућег формата, да ли су унети подаци у скупу дозвољених вредности за део формата коме припадају и друго.

Изузеци: нема

Услови који морају бити задовољени после извршавања: Креира се извештај о валидности библиографског записа

Случај коришћења: Приказ записа у различитим форматима

Кратак опис: Делови записа или цео запис приказују се на екрану у изабраном формату

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Запис је припремљен за приказивање

Опис: На екрану се приказује цео или део записа за публикацију која се обрађује. Корисник бира у ком формату жели да му се прикаже запис, а понуђени формати су унапред дефинисани.

Изузеци:

Услови који морају бити задовољени после извршавања: Запис је приказан у изабраном формату

Случај коришћења: Учитавање записа

Кратак опис: Постојећи библиографски запис учитава се у едитор за обраду библиографске грађе

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Изабран је запис за учитавање

Опис: У едитор за обраду библиографске грађе учитава се постојећи библиографски запис ради модификације или креирања новог записа на основу постојећег.

Изузеци: нема

Услови који морају бити задовољени после извршавања: Запис је учитан у едитор и приказан на екрану

Случај коришћења: Снимање библиографског записа

Кратак опис: Врши се снимање записа ради његовог трајног чувања

Учесници: Библиотекар

Услови који морају бити задовољени пре извршавања: Меморијска структура записа је спремна за снимање у неком од формата за репрезентацију записа

Опис: На захтев корисника, врши се трајно снимање записа. Пре него што се запис сачува врши се валидација целог записа, запис ће бити сачуван само уколико је валидан.

Изузеци: нема

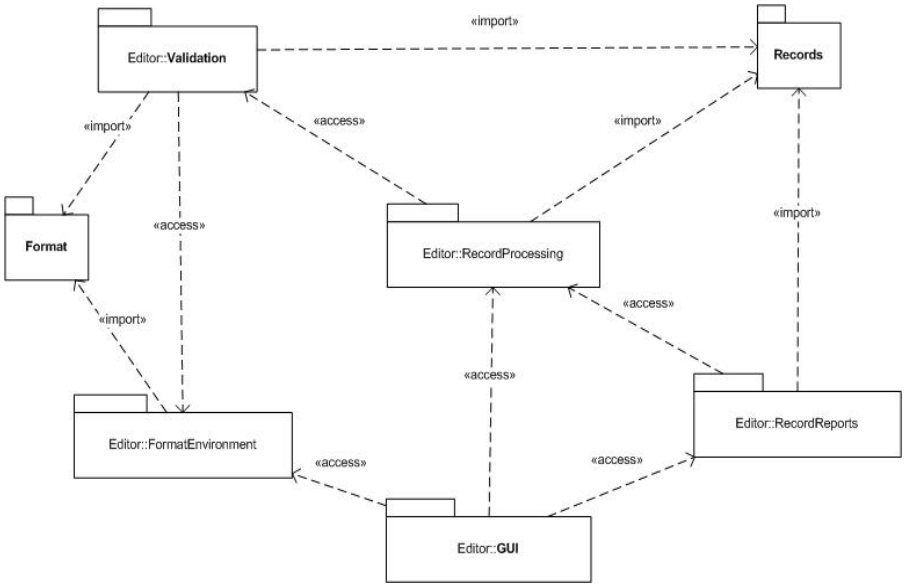
Услови који морају бити задовољени после извршавања: Запис је трајно сачуван или је послата порука о грешци у случају невалидности записа

3.2 Статички модел система за обраду библиографске грађе

3.2.1 Дијаграм пакета

Дијаграм пакета система за обраду библиографске грађе приказан је на слици 3.2.

Пакет *Records* садржи класе којима се представља меморијска структура библиографског записа. Пакет *Format* садржи класе за опис меморијске структуре самог формата. Остали пакети приказани на дијаграму представљају угњеждене пакете пакета *Editor*.



Слика 3.2 *Дијаграм пакета едитора за обраду библиографске грађе*

Пакет *RecordProcessing* је централни пакет у процесу обраде библиографске грађе јер садржи инстанцу меморијске репрезентације библиографског записа који се обрађује и приказује на екранској форми едитора. Из разлога што овај пакет користи меморијску структуру записа он увози садржај пакета *Records*. У процесу каталогизације врши се обрада једног записа, те је у сваком моменту потребно чувати тренутно стање у ком се запис налази. Овај податак налазиће се у оквиру пакета *RecordProcessing* као појава одговарајуће класе из пакета *Records*. Поред података о запису пакет *RecordProcessing* садржи и класе за обраду меморијске структуре записа у коју спадају учитавање записа у меморијску структуру, као и управљање додавањем, модификацијом и уклањањем података из меморијске структуре записа. Запис се учитатава у

меморијску структуру ради модификације или креирања новог записа на основу постојећег, и може се учитати из различитих извора, на пример из апликације која врши претраживање удаљених база података, из локалне базе података или из фајл система.

Пакет *FormatEnvironment* садржи класе које управљају меморијском структуром формата која се увози из пакета *Format*. Приликом обраде записа постоји подскуп формата чији се елементи могу селектовати ради уноса података, али је омогућена и модификација овог подскупа у виду додавања нових елемената или уклањања постојећих. Скуп свих елемената формата који се могу селектовати ради уноса података налазиће се у оквиру овог пакета као појава одговарајуће класе пакета *Format*. У функционалност која је моделирана овим пакетом спадају и процес складиштења података о подскупу формата ради његовог поновног коришћења као и учитавање већ креираног подскупа формата.

Пакетом *Validation* моделирана је валидација записа. Овај пакет садржи класе чијим операцијама је могуће проверити да ли је запис креиран у складу са правилима које прописује библиотечки стандард по ком се врши каталогизација. Операцијама које врше валидацију записа приступа се из пакета *RecordProcessing*, јер се валидација врши у току обраде записа. На овај начин омогућено је да се поруке о невалидности унетих података кориснику пошаљу у току саме обраде. Валидација се врши на основу меморијске репрезентације записа из пакета *RecordProcessing* и меморијске репрезентације формата из пакета *FormatEnvironment*. Како се у процесу валидације користе меморијске структуре записа и формата овај пакет увози садржаје пакета којима су моделиране ове структуре.

Пакет *GUI* садржи класе за креирање оног дела апликације који се односи на изглед едитора и функционалност графичких елемената едитора. Овај пакет приступа пакету *FormatEnvironment* ради преузимања података о подскупу формата који се у облику стабла приказује кориснику, као и пакету *RecordProcessing* ради преузимања записа који се приказује кориснику на екранској форми едитора.

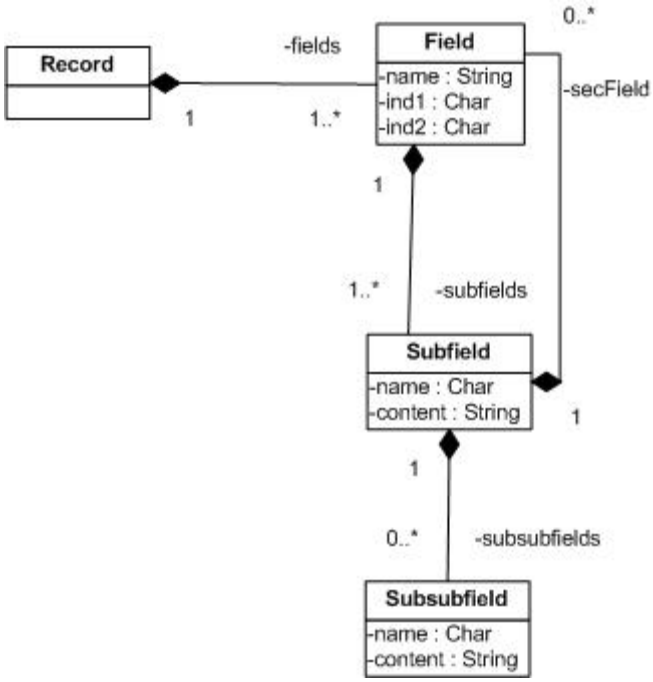
Приликом обраде записа јавља се потреба де се исправност унетих података провери путем приказа у форми листића или некој другој форми која је кориснику читљива. Процес формирања и приказивања записа у различитим облицима моделиран је пакетом *RecordReports*. Овом пакету приступаће се из пакета *GUI* јер се операције за приказ листића извршавају на захтев корисника. Као улазни параметар у процесу приказивања записа јавља се запис који се тренутно обрађује и преузима се из пакета *RecordProcessing* што је моделирано везом са стереотипом *access* између поменута два пакета. Подаци за креирање приказа записа преузимају се из меморијске структуре записа и постоји потреба за

коришћењем операција над овом структуром због чега пакет *RecordReports* увози садржај пакета *Records*.

3.2.2 Дијаграм класа пакета *Records*

Пакет *Records* садржи класе којима се описује меморијска структура за репрезентацију библиографског записа. Дијаграм класа пакета *Records* зависи од структуре записа, односно од библиотечког формата по ком се врши каталогизација. У наставку ће бити дат дијаграм који описује објектни модел библиографског записа по YUMARC формату

Основна улога објектних модела библиографских записа јесте репрезентација података о запису у апликацији. Подаци о запису се могу учитати у објектни модел из XML докумената чија спецификација је дата у другом поглављу ове тезе. Такође, предвиђено је да се подаци о запису складиште у те XML документе. Да би се извршило правилно мапирање података из XML докумената на објектни модел, и обрнуто, потребно је да објектни модел одговара моделу XML шеме према којој се креирају XML библиографски записи.



Слика 3.3 *Дијаграм класа објектног модела библиографског записа по YUMARC стандарду*

На слици 3.3 дат је дијаграм класа објектног модела библиографског записа по YUMARC стандарду. Овај дијаграм класа

креиран је на основу XML шеме библиографског записа по YUMARC формату која је описана у одељку 2.1.4.

Класом *Record* моделиран је библиографски запис. Он садржи вишеструке појаве класе *Field* којом је моделирано поље записа. Поље садржи више потпоља која су моделирана класом *Subfield*. Ова класа може садржати и појаве класе *Field* које се односе на секундарна поље која се јављa у оквиру посматраног потпоља и може садржати појаве класе *Subsubfield* којима је моделирано потпотпоље YUMARC формата.

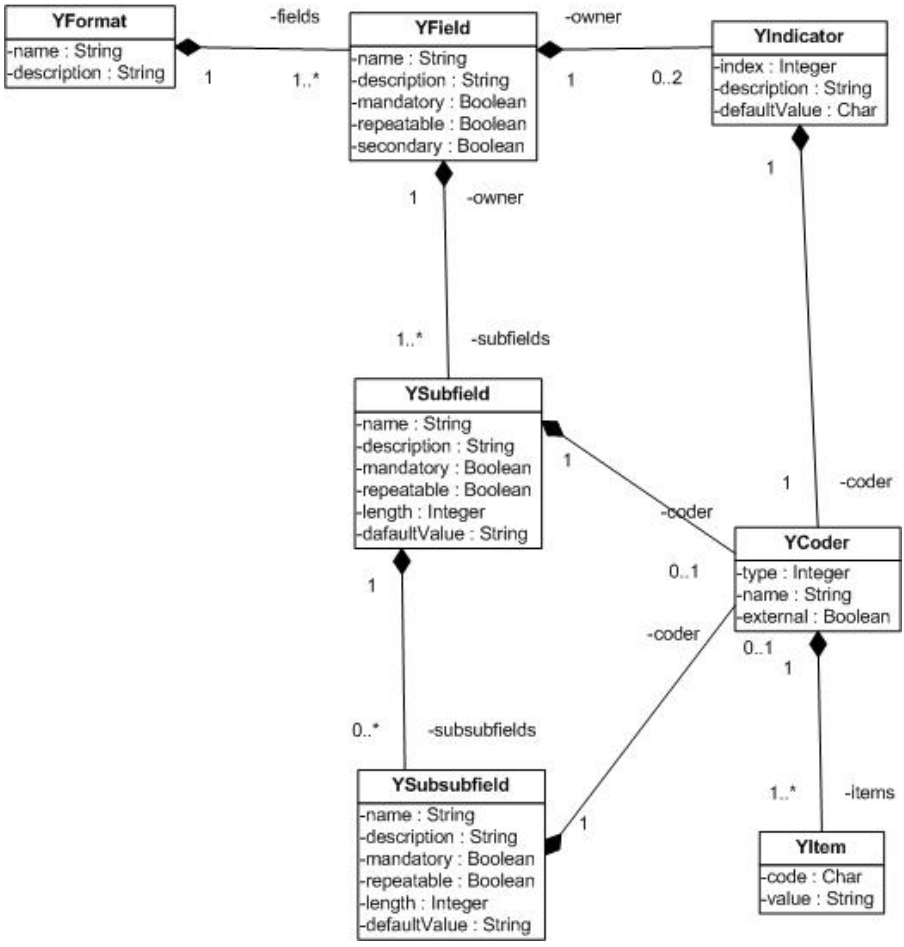
3.2.3 Дијаграм класа пакета *Format*

Пакет *Format* садржи класе које моделирају меморијску структуру за опис података о библиотечком формату или подскупу библиотечког формата по ком се врши обрада библиографске грађе у едитору. У наставку ће бити дат дијаграм класа библиотечког формата YUMARC.

Основна улога објектних модела библиотечких формата је репрезентација података о формату у апликацији. На основу информација које садрже ови објектни модели врши се контрола уноса података у библиографски запис. Подаци о библиотечком формату учитавају се у објектни модел из XML докумената чија спецификација је дата у другом поглављу ове тезе. Исто као у случају библиографских записа, да би се извршило правилно мапирање података из XML докумената на објектни модел потребно је да ови објектни модели буду крерани на основу XML шема из чијих инстанци ће се учитавати подаци.

На слици 3.4 приказан је дијаграм класа објектног модела YUMARC формата који је креиран на основу XML шеме описане у одељку 2.1.1.

Класа *YFormat* описује модел целокупног формата или његовог подскупа и садржи вишеструке појаве класе *YField* којом је моделирано поље формата. Поље у формату се састоји од низа потпоља моделираних класом *YSubfield* и највише два индикатора која су моделирана класом *YIndicator*. Потпоља могу садржати још један ниво подструктуре – потпотпоља која су моделирана класом *YSubsubfield*. Класом Y*Coder* моделирани су шифарници формата, а класом *YItem* ставка у шифарнику.



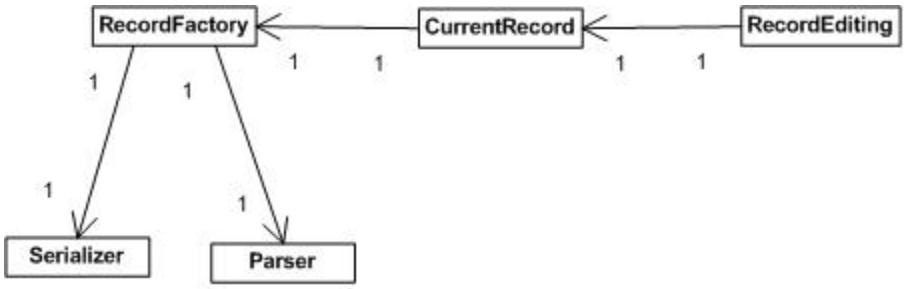
Слика 3.4 *Дијаграм класа објектог модела YUMARC формата*

3.2.4 Дијаграм класа пакета *RecordProcessing*

Дијаграм класа које описују обраду меморијске структуре библиографског записа приказан је на слици 3.5.

Класа *RecordFactory* садржи операције за учитавање података у меморијску структуру записа, као и трансформацију меморијске структуре у текстуални облик представљања записа. Подаци о запису могу се учитати из различитих формата, а као најчешћи формат у ком се записи трајно чувају је XML. Класа *RecordFactory* користи операције класе *Parser* којима се врши парсирање било XML документа или неког другог облика репрезентације записа. Операцијама класе *Serializer* могуће је извршити процес супротан учитавању записа у меморијску структуру. Наиме, на основу објектног модела библиографског записа креирају се различити

облици представљања записа ради трајног чувања или преноса података о запису до других апликација.



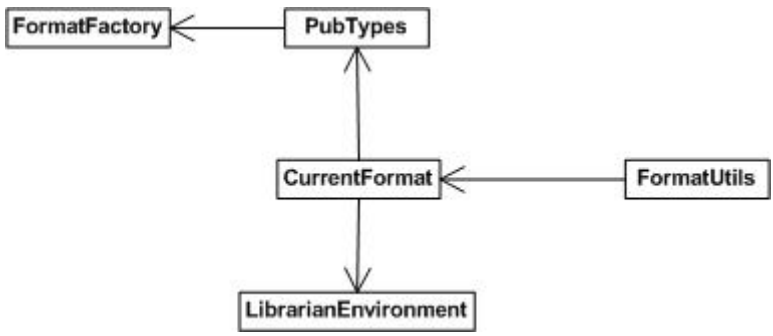
Слика 3.5 *Дијаграм класа пакета RecordProcessing*

Класа *CurrentRecord* садржи све податке који су везани за библиографски запис који се тренутно обрађује као и податке неопходне за сам процес обраде меморијске структуре записа. Он садржи инстанцу записа, као и податке који су везани за саму обраду, а ту спадају тренутно селектовани елемент записа, елемент записа који је последњи модификован и друго.

Класа *RecordEditing* моделира едитирање записа на нивоу меморијске репрезентације записа.

3.2.5 Дијаграм класа пакета *FormatEnvironment*

На слици 3.6 представљен је дијаграм класа за манипулацију меморијском структуром библиотечког формата за обраду библиографске грађе.



Слика 3.6 *Дијаграм класа пакета FormatEnvironment*

Класа *FormatFactory* садржи операције за учитавање података о целом библиотечком формату у меморијску структуру. Подаци о формату који се учитавају могу бити представљени у облику XML документа чијим

парсирањем се врши учитавање података. Међутим, подаци о формату се могу учитавати и из табела релационе базе података или из неког другог начина репрезентације.

Класа *PubTypes* садржи операције за издвајање подскупа формата који се односи на конкретан тип публикације. Ова класа користи операције класе *FormatFactory* за добијање меморијске репрезентације целог формата из кога се на основу спецификације типова публикације која је дата у посебном XML документу издваја објектни модел подскупа формата који садржи поља и потпоља која према библиотечком стандарду припадају одређеном типу публикације.

Класа *LibrarianEnvironment* моделира окружење библиотекара. Овом класом дефинисани су типови обраде за библиотекара који дефинишу скуп поља и потпоља на основу којих се креира подскуп библиотечког формата по ком се врши обрада.

Класа *CurrentFormat* садржи меморијску репрезентацију подскупа библиотечког формата који се у облику стабла приказује кориснику на едитору. Под подскупом формата подразумева се скуп елемената формата дефинисаних библиотечким стандардом које приликом обраде записа корисник селектује ради уноси вредности. Он може изабрати само оне делове формата који су обухваћени подскупом формата из класе *CurrentFormat*. Такође, ова класа обезбеђује додатне функционалности које се односе на припрему скупа поља и потпоља приликом додавања недостајућих елемената у подскуп формата, затим на припрему шифарника који се налазе у оквиру објектног модела библиотечког формата, као и прибављање осталих података за контролу уноса података у запис.

У процесу креирања почетног подскупа формата ова класа преузима податке о типу обраде пријављеног библиотекара из класе *LibrarianEnvironment*. У случају када треба додати неки елемент формата класа *CurrentFormat* приступа класи *PubTypes* ради преузимања додатних елемената формата који су дефинисани типом публикације који се обрађује.

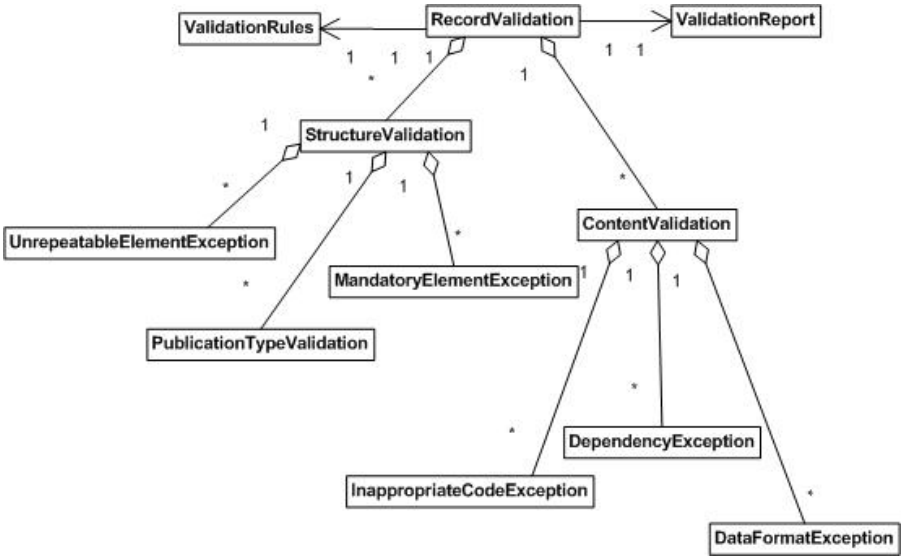
Класа *FormatUtils* је помоћна класа за обраду објектног модела подскупа формата. Она садржи помоћне операције за обраду меморијске структуре за репрезентацију формата.

3.2.6 Дијаграм класа пакета *Validation*

Валидација записа представља процес утврђивања да ли библиографски запис задовољава правила за обраду библиографске грађе која су прописана изабраним библиотечким стандардом. Дијаграм класа који описује валидацију записа дат је на слици 3.7.

Основна класа за валидацију је класа *RecordValidation*. Валидација се врши на основу правила за валидацију дефинисаних класом *ValidationRules*, а након валидације креира се извештај о валидности записа помоћу операција класе *ValidationReport*. Уколико запис није валидан у овом изваштају биће наведени сви делови записа који нарушавају валидност као и разлози због којих они нису у складу са стандардом.

Процес валидације може се поделити на два дела, контрола структуре записа и контрола садржаја записа. У складу са овим, класа *RecordValidation* представља агрегацију класа *StructureValidation* за валидацију структуре и *ContentValidation* за валидацију садржаја записа.



Слика 3.7 *Дијаграм класа пакета Validation*

Невалидност структуре записа може настати у случају да се елемент записа који не може бити поновљив јавља више пута у запису што је контролисано класом *UnrepeatableElemenException*. Други разлог невалидности структуре који се односи на непојављивање обавезних елемената записа регулисан је класом *MandatoryElementException*. Такође, под контролом структуре записа подразумева се и провера усклађености записа са типом публикације за коју се креира запис. Овакав тип контроле моделиран је класом *PublicationTypeValidation*.

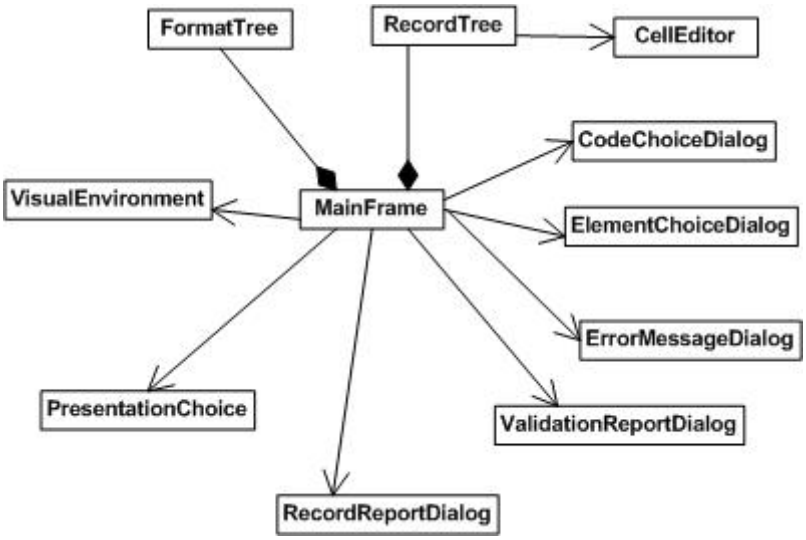
Контрола садржаја записа може се поделити на контролу шифара, контролу формата података и тзв. унакрсну контролу. Контрола шифара моделирана је класом *InappropriateCodeException* која ће генерисати поруку о грешци уколико унета шифра није у скупу дозвољених

вредности за посматрани елемент записа. Под контролом формата података подрезумева се провера дужине податка за потпоља или контролна поља која имају дефинисану дозвољену дужину, провера да ли унете вредности задовољавају унапред утврђене критеријуме (на пример да ли је унети податак број или да ли унети податак има формат године) и слично. Контрола формата података моделирана је класом *DataFormatException*. Класом *DependancyException* представљена је унакрсна контрола којом се утврђује да ли је садржај ознаке слога, одређених потпоља, контролних поља или индикатора одговарајући у односу на структуру или садржај других поља, потпоља или индикатора.

Веза агрегације између појединачних делова валидације записа моделира случај да се класе са значењем *део* у овој вези могу користити и независно од класе која представља *целину*. Потреба за оваквом везом јавља се због тога што ће се процес валидације вршити и постепено у процесу обраде те се јавља потреба да се класе које моделирају делове валидације користе независно од класе која моделира валидацију целог записа.

3.2.7 Дијаграм класа пакета *GUI*

Дијаграм класа који представља модел корисничког интерфејса едитора за обраду библиографске грађе приказан је на слици 3.8.



Слика 3.8 *Дијаграм класа пакета GUI*

Класа *MainFrame* представља основну класу за креирање изгледа едитора и дефинисање функционалности графичких елемената на едитору. Ова класа представља композицију две класе, класе *FormatTree* која моделира графички приказ стабла за опис библиотечког формата и класе *RecordTree* која представља графички приказ библиографског записа који се обрађује.

Класа *PresentationChoice* моделира графички приказ и функционалност избора врсте листића за приказивање записа.

Класом *FormatTree* описује се графичка репрезентација стабла за опис библиотечког формата или његовог подскупа. Ова класа моделира графичку структуру стабла формата као и операције над овом структуром. У операције над структуром стабла формата спадају приказ елемената формата у стаблу, кретање по структури стабла, додавање односно уклањање поља и потпоља у стаблу и селекција оних елемената у стаблу за које се врши унос података.

Класа *RecordTree* описује структуру која служи за графички приказ библиографског записа који се обрађује. Запис је на екранској форми едитора представљен у облику стабла у ком су подаци о запису представљени у *full* формату. У стаблу записа могуће је селектовати елемент записа ради уноса или модификације библиотечких података који се врши у форми за унос која је моделирана класом *CellEditor*. Поред тога, над стаблом формата је могуће вршити операције које се односе на промену структуре записа, као што су промена редослета елемената у запису или брисање елемената записа.

Класа *PresentationChoice* моделира графички приказ свих врста листића или других облика у којима се запис може приказивати као и избор врсте листића за приказ. Након што корисник, помоћу операција ове класе одабере врсту листића, запис се шаље софтверској компоненти која формира листић и тако формиран листић се, такође операцијама ове класе приказује на едитору.

Класа *VisualEnvironment* описује особине графичког окружења и путем операција ове класе корисник ће моћи да подеси графичко окружење по сопственом избору, ту спадају одабир боје и величине појединих графичких делова едитора као и величине и врсте слова на едитору.

Остале класе служе за управљање помоћним прозорима и имају следећу функционалност:

- *CodeChoiceDialog* - избор шифре из шифарника,
- *ErrorMessageDialog* - поруке о грешци приликом уноса,

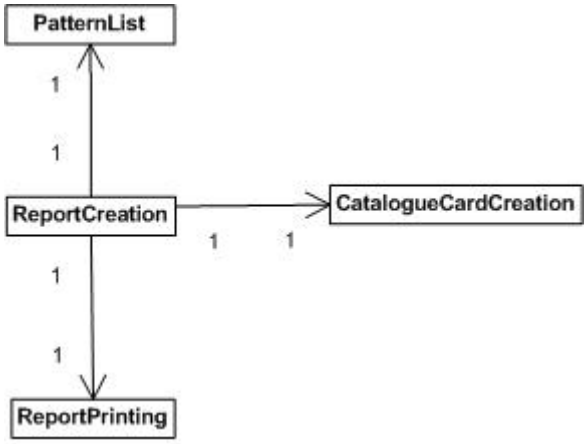
- *ElementChoiceDialog* - избор елемента за додавање у стабло формата,
- *ValidationReportDialog* - приказ извештаја валидације библиографског записа и
- *RecordReportDialog* - приказ записа у форми листића или другој форми за приказ записа.

3.2.8 Дијаграм класа пакета *RecordReports*

Дијаграм класа пакета који управља формирањем различитих облика приказивања записа приказан је на слици 3.9.

Процесом формирања каталошких листића или других начина приказа записа управља класа *ReportCreation*.

Поред форме листића запис ће моћи да се приказује у разним формама који су кориснику читљиви или погодни за контролу унетих података. Класа *PatternList* моделира листу свих постојећих облика за приказивање записа. Из ове листе корисник ће моћи да изабере облик за приказивање записа.



Слика 3.9 *Дијаграм класа пакета RecordReports*

Класа *CatalogueCardCreation* репрезентује формирање каталошког листића на основу изабране форме листића путем позивања независне софтверске компоненте.

Класа *ReportPrinting* обезбеђује штампање меморијске структуре записа у изабраном облику.

3.3 Динамички модел система за обраду библиографске грађе

Основна функционалност коју едитор за обраду библиографске грађе треба да подржи јесте креирање библиографског записа према правилима које прописује изабрани библиотечки стандард. Креирање библиографског записа састоји се од низа операција које се могу поделити у две групе:

- креирање структуре записа и
- унос библиографских података.

Под креирањем структуре библиографских записа подразумевају се операције додавања елемената у запис, као и промена редоследа или брисање елемената записа. Приликом додавања нових елемената у запис потребно је извршити контролу структуре библиографског записа који се обрађује, тачније њену усклађеност са правилима изабраног стандарда.

Под уносом библиографских података подразумева се операција уноса садржаја у конкретан елемент библиографског записа од стране корисника, библиотекара. Приликом уноса података врше се контроле унетог садржаја.

Систематизација контрола библиогрефских записа, као и динимички модел система за контролу библиографских записа на примеру UNIMARC формата дат је у [Dimić07b].

Едитор за обраду библиографске грађе, приказан у овој тези предвиђен је да подржи обраду, а самим тим и контролу уноса података за различите библиотечке формате. У овом одељку су дати дијаграми активности који илуструју динамички модел система за контролу библиографских записа по било ком формату.

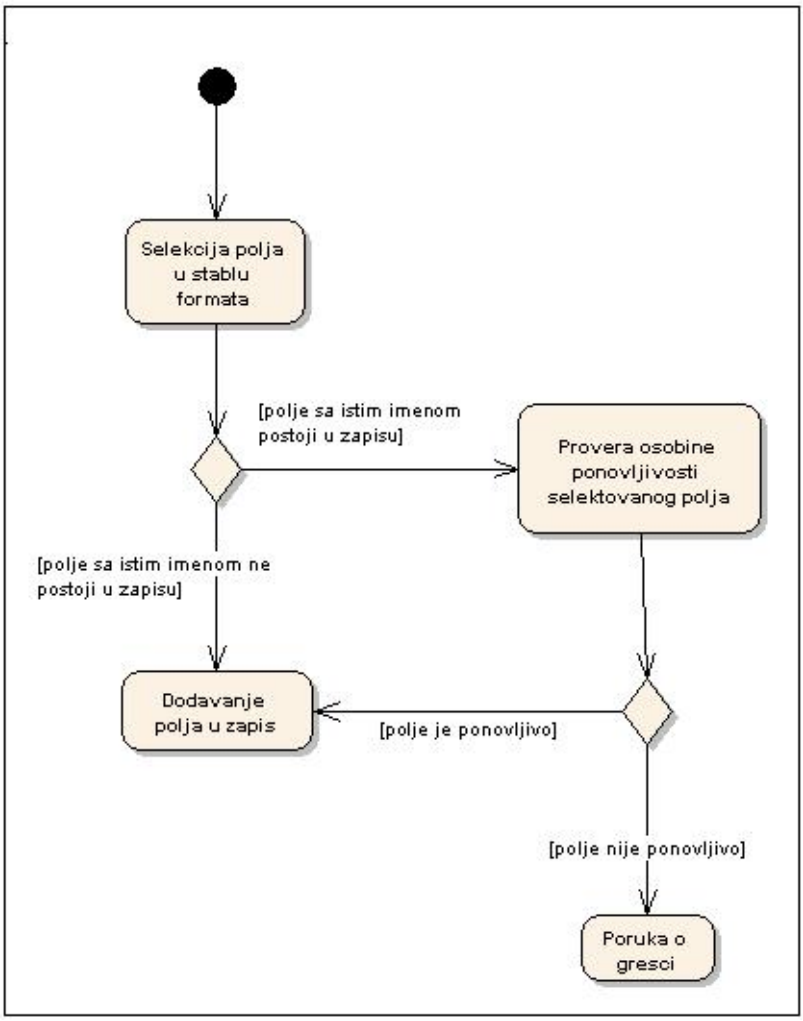
Контроле библиографских записа које су описане у овом одељку врше се над објектним моделом библиографског записа по изабраном формату. Подаци на основу којих се врши контрола смештени су у објектном моделу изабраног библиотечког формата.

Контрола креирања структуре библиографских записа описана је на примеру додавања новог поља у запис. Приликом додавања осталих елемената у запис примењују се исте акције као и приликом додавања поља.

Контрола уноса библиографских података дата је на примеру уноса садржаја потпоља. Унос садржаја у неки други елемент записа врши се по сличном поступку.

3.3.1 Додавање поља у запис

На слици 3.10 дат је дијаграм активности контроле структуре библиографског записа која се врши приликом додавања поља у запис.



Слика 3.10 *Дијаграм активности контроле додавања поља у запис*

Додавање поља у запис започиње селекцијом поља у стаблу формата. Стабло формата се креира на основу објектног модела формата, или његовог подскупа по ком се врши каталогизација. На овај начин онемогућен је унос елемента који није дефинисан форматом.

Пре додавања поља у запис потребно је проверити да ли би овим додавањем била нарушена валидност структуре записа у односу на

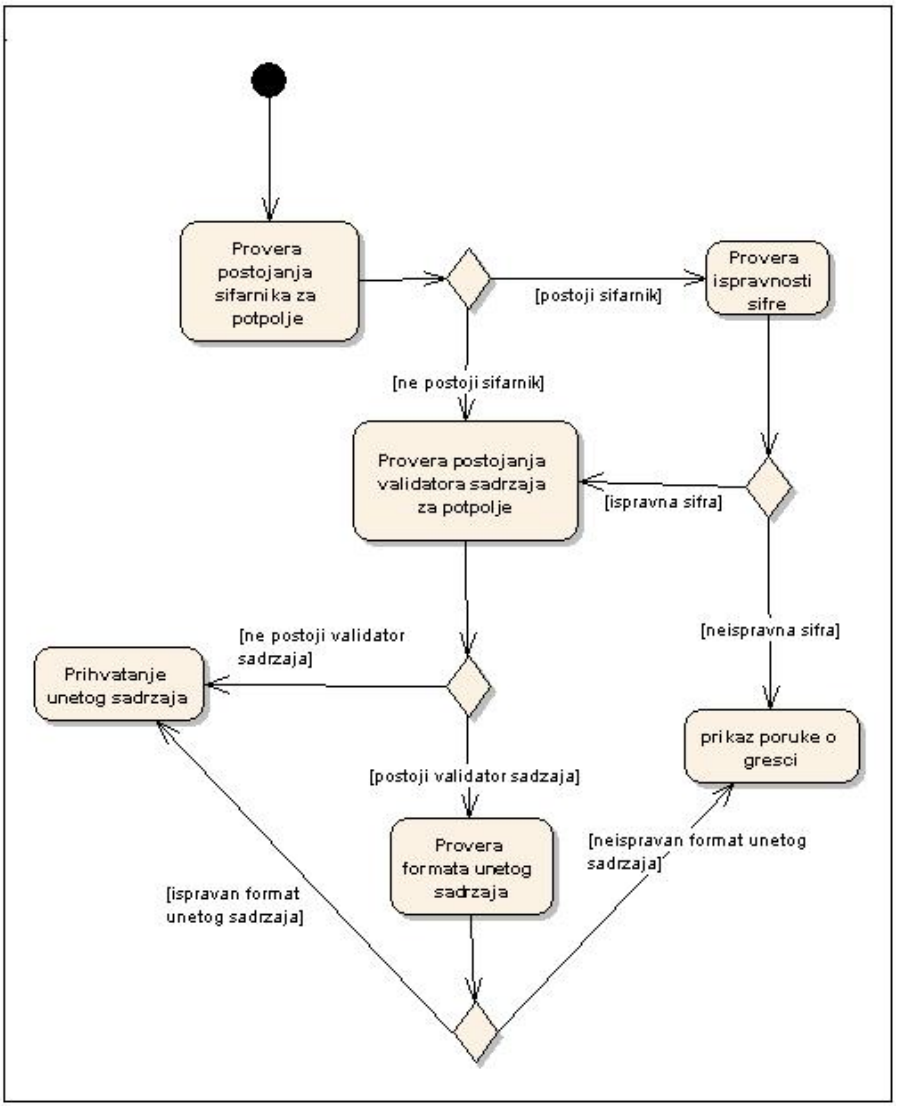
изабрани библиотечки формат. То нарушавање би се десило ако би се додало поље које већ постоји у запису, а по дефиницији формата нема особину поновљивости.

Ако је селектовано поље које не постоји у запису, није потребно вршити никакве контроле, поље се једноставно додаје у запис. Уколико је селектовано поље које већ постоји у запису проверава се да ли то поље има особину поновљивости. Ако поље нема особину поновљивости, његовим додавањем нарушила би се валидност структуре записа, због чега се шаље порука о грешци и поље се не додаје у запис. Ако се утврди да поље има особину поновљивости, поље се додаје у запис.

3.3.2 Унос садржаја потпоља

Приликом уноса библиографских података у запис врши се контрола садржаја која се уноси. На слици 3.11 дат је дијаграм активности који описује процес контроле садржаја који се уноси у потпоље записа. Овај процес ће се извршити након уноса податка од стране корисника пре прихватања унетог текста као садржаја потпоља.

Приликом уноса садржаја потпоља прво се проверава да ли постоји шифарник везан за дато потпоље. Уколико постоји шифарник проверава се да ли је унета вредност у скупу дозвољених шифара за потпоље. Ако шифра није добра генерише се одговарајућа порука. Ако је унета исправна шифра или не постоји шифарник везан за потпоља проверава се да ли постоји валидатор садржаја за потпоље, односно да ли то потпоље прихвата садржај тачно дефинисаног формата. Ако не постоји валидатор садржаја за потпоље, завршена је контрола уноса садржаја и унета вредност се прихвата као садржај потпоља. Уколико постоји валидатор садржаја за потпоље врши се провера исправности формата унетог садржаја. Уколико садржај није унет у исправном формату генерише се порука о грешци. У случају да је садржај доброг формата он се прихвата као садржај потпоља.



Слика 3.11 *Дијаграм активности контроле уноса садржаја потпоља*

Поглавље 4

Имплементација XML едитора за обраду библиографске грађе

Ово поглавље садржи опис основних концепата имплементације XML едитора за обраду библиографске грађе. Имплементација је извршена на основу спецификације која је представљена у трећем поглављу. У овом одељку дат је опис софтверских алата коришћених приликом имплементације као и приликом интеграције едитора у четврту верзију софтверског система БИСИС.

4.1 Софтверско окружење

Едитор за обраду библиографске грађе који је описан у овој монографији базиран је на XML технологијама, а имплементација је извршена у програмском језику *Java*. Развој апликације извршен је у програмском пакету *Eclipse*. За интеграцију у библиотечки софтверски систем БИСИС, која се односи на процес складиштења и индексирања библиографских записа коришћен је текст сервер *Lucene* и *MySQL* релациона база података.

Програмски језик *Java* [Java] је *open source* производ фирме *Sun Microsystems*. То је објектно оријентисани програмски језик, специјално дизајниран да што мање зависи од специфичних карактеристика рачунарског система. *Java* омогућава да се једном написан и преведен програм може извршавати на различитим платформама које је подржавају. Коришћена верзија програмског језика *Java* је 1.6.0.

У оквиру *Java* платформе реализована је специјална библиотека класа која омогућава ефикасну имплементацију корисничког интерфејса, то је пакет *swing*. Пакет *swing* садржи богат скуп компонената за креирање корисничког интерфејса. Коришћењем класа из овог пакета могуће је реализовати кориснички интерфејс који подржава све савремене визуелне концепте пословних апликација.

Апликација описана у овој тези базира се на XML технологијама. За спецификацију XML докумената коришћен је XMLSchema језик [XMLSchema]. За обраду XML докумената у апликацији коришћен је *Simple API for XML (SAX)* [SAX] имплементиран у *Java* окружењу.

Eclipse [Eclipse] је *open source* пројекат који представља развојно окружење за израду софтвера. Овај алат обезбеђује рад са различитим

програмским језицима и на различитим платформама. *Eclipse* обезбеђује *plug-in* оквир за интеграцију додатних софтверских алата. Последња актуелна верзија, која је и коришћена приликом имплементације едитора описаног у овој тези је 3.2.2.

Lucene [Lucene] је софтвер за индексирање и претраживање података (Information Retrieval (IR) Library). То је бесплатан, *open-source*, пројекат имплементиран у *Java*-и, у свом зачетку припадао је популарној фамилији *Apache Jakarta* [Jakarta] пројеката, али због своје популарности постаје самосталан пројекат *Apache Foundation*-а [Apache]. Као такав, тренутно, и већ неколико година је најпопуларнији бесплатни *Java* (IR Library) претраживач информација.

MySQL [MySQL] је *open source* систем за управљање релационим базама података добрих перформанси, високе поузданости и једноставан је за употребу. Може се покренути на различитим платформама, као што су Linux, Windows, OS/X, HP-UX, и друге.

4.2 Главна екранска форма едитора

Изглед главне екранске форме XML едитора за обраду библиографске грађе дат је на слици 5.9.

Главна екранска форма едитора састоји се из три основна дела:

- стабло библиотечког формата (лева страна екранске форме),
- стабло библиографског записа (десна страна екранске форме) и
- панел са дугмадима за брзи приступ командама (горња страна екранске форме).

Екранска форма едитора приказана на слици 5.9 имплементира целокупну функционалност система за обраду библиографске грађе која је описана дијаграмима случајева коришћења у одељку 3.1. У наставку ће бити описана имплементација ових случајева коришћена на главној екранској форми едитора.

Случај коришћења *Подешевање окружења* имплементиран је над стаблом библиотечког формата, које представља графичку репрезентацију окружења које се односи на тренутни подскуп библиотечког формата над којим се врши обрада библиографске грађе. Приликом првог покретања едитора стабло библиотечког формата формира се на основу подразумеваног типа обраде за пријављеног библиотекара. Дугме са иконицом  омогућава кориснику да промени тип обраде при чему се након изабраног неког другог типа обраде ажурира стабло библиотечког формата.

Појединa дугмад на панелу за брзи приступ командама који се

налази у горњем делу екранске форме едитора представљају имплементацију неких од случајева коришћења. Тако дугме са иконицом  реализује случај коришћења *Ажурирање стабла библиотечког формата* јер ово дугме представља пречицу за команду додавања елемената у стабло формата.

Дугме са иконицом  реализује случај коришћења *Снимање записа*. Имплементација функционалности која се односи на снимање записа представља складиштење меморијске репрезентације записа у XML документ, а даље складиштење ових докумената зависи од софтверског система у који се едитор интегрише.

Дугме са иконицом  реализује део случаја коришћења *Валидација записа* који се односи на валидацију целог записа. Поред тога, део случаја коришћења *Валидација записа* реализује се као део уноса података и он подразумева проверу појединачних података у току самог уноса.

Дугме са иконицом  реализује случај коришћења *Приказ записа у различитим форматима*. Функционалност ове команде реализује се коришћењем посебног подсистема за генерисање каталошких листића. У едитору се она реализује тако што се увек отвара одређени тип листића који је дефинисан као подрезумевајући за библиотеку. Поред тога, кориснику се нуди листа свих дефинисаних врста листића из које он може да одабере одговарајући листић. Листић се генерише у облику HTML странице у посебној софтверској компоненти којој се прослеђује тип листића.

Случај коришћења *Креирање библиографског записа* реализован је над графичком компонентом стабла библиографског записа. Над овим стаблом имплементирана је целокупна функционалност за обраду структуре записа, ту спадају додавање елемената, брисање елемената, промена редоследа елемената записа. Поред тога над овом графичком компонентом имплементиран је и случај коришћења *Унос података* у виду отварања посебног текстуалног едитора приликом селекције дела записа у који се уноси садржај. Изглед овог текстуалног едитора реализован је различито за унос шифрираних и нешифрираних садржаја. Приликом обраде стабла записа, чиме се врши креирање библиографског записа врши се и део функционалности случаја коришћења *Валидација записа* који се односи на појединачне контроле елемената записа.

Случај коришћења *Учитавање записа* представља функционалност отварања екранске форме едиотара за обраду већ постојећег записа, било ради модификације или креирања новог записа на основу постојећег. Након учитавања записа, стабло записа едитора је креирано на основу учитаног записа и над њим се врше потребне модификације.

Функционалност овог XML едитора која се односи на учитавање записа своди се на учитавање XML документа библиографског записа у објектни модел записа на основу кога се креира графичка репрезентација тог записа у облику стабла. Начин на који је креиран XML документ који се учитава у објектни модел зависи од система у који се едитор интегрише, и то најчешће оног дела који се односи на претраживање неке колекције библиографских записа.

4.3 Објектни модел библиографског записа

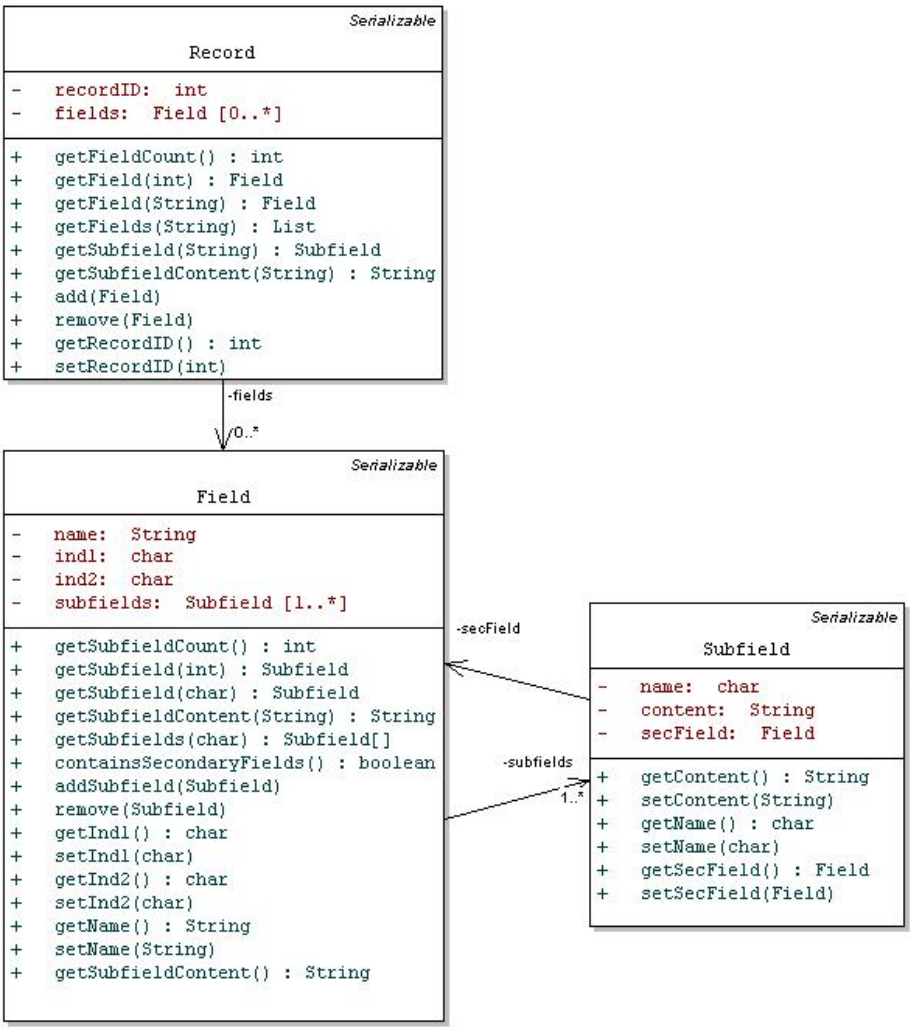
Све операције над библиографским записом у едитору извршавају се над објектним моделом библиографског записа који је имплементиран на основу дијаграма класа који је приказан на слици 3.3. Те операције се састоје од промене структуре и садржаја записа а реализују се коришћењем *set* и *get* метода објектног модела записа. Поред ових *set* и *get* метода за приступ садржајима атрибута класа, класе објектног модела садрже и операције над колекцијама података, као што су додавање елемената у листу, уклањање елемента из листе, сортирање листе и др.

На слици 4.1 приказана је спецификација објектног модела библиогарфског записа при чему су за све класе наведени атрибути и операције.

Класа *Record* моделира библиографски запис. Запис је одређен својим бројем који је садржај атрибута *recordId*. Запис садржи и вишеструку појаву поља записа која је представљена атрибутом *fields*. Скуп метода класе *Record* састоје се од *set* и *get* метода за приступ атрибуту *recordId* и метода за обраду скупа поља записа. На пример, приликом додавања поља у запис позива се метода *add(Field)* а за уклањање поља из записа користи се метода *remove(Field)*, мотоди *getField(String)* прослеђује се назив поља и она враћа прво поље у запису са тим називом. Метода *getFields(String)* враћа листу поља чија имана одговарју улазном параметру.

Класа *Field* представља поље записа. Поље је одређено својим именом које је представљено атрибутом *name*. Поред тога поље садржи и атрибуте *ind1* и *ind2* које представљају вредности индикатора и атрибут *subfields* који представља листу потпоља.

Поред *set* и *get* метода за приступ садржајима атрибута ова класа садржи и методе за обраду скупа потпоља. Метода *getSubfield(char)* враћа потпоље са чији назив одговара улазном параметру. Метода *addSubfield(Subfield)* додаје потпоље у листу, а операције *remove(Subfield)* уклања потпоље из листе.



Слика 4.1 *Објектни модел записа*

4.4 Имплементација обраде меморијске структуре библиографског записа

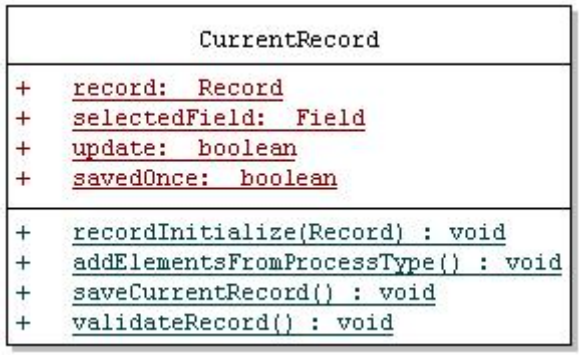
Статички модел обраде објектног модела записа у едитору дат је у пакету *RecordProcessing* чији је дијаграм класа приказан на слици 3.5.

4.4.1 Имплементација класе *CurrentRecord*

На слици 4.2 приказана је класа *CurrentRecord*. Атрибутом *record* представљен је запис које се тренутно обрађује у едитору. Све операције

које корисник изврши над записом у едитору врше се над овом појавом записа. Такође, на основу стања овог објекта ажурира се приказ записа у едитору.

Приликом позива операције за додавање потпоља, потпоље се додаје у поље које је селектовано у стаблу записа. Атрибут *selectedField* типа *Field* представља инстанцу тренутно селектованог поља у запису. Класа *CurrentRecord* садржи и логички атрибут *update* који одређује да ли се у едитору креира нови запис или се модификује постојећи као и атрибут *savedOnce* које даје информацију о томе да ли је запис који се тренутно обрађује већ снимљен.



Слика 4.2 *Класа CurrentRecord*

Класа *CurrentRecord* садржи операције које се односе на иницијализацију и складиштење записа, као и проверу валидности записа. Операција *recordInitialize(Record)* врши иницијализацију записа и позива се приликом покретања едитора. Ова операција креира поље *record* на основу прослеђене вредности и окружења пријављеног библиотекара. У случају да се едитор покреће ради креирања потпуно новог записа као параметар ове операције прослеђује се *null* вредност и тада се запис иницијализује на основу окружења пријављеног библиотекара. Уколико се у едитору отвара постојећи запис, он ће се проследити као параметар ове операције, а позивом помоћне операције *addElementsFromProcessType*() у запис се додају и елементи који су дефинисани окружењем библиотекара.

Операција за снимање записа је *saveCurrentRecord()* која врши складиштење објекта *record*. Ова операција прослеђује изузетак у случају невалидности записа.

Први корак у снимању записа је провера валидности записа позивом операције *validateRecord()*. Операција *validateRecord()* на основу података из формата проверава валидност записа који је представљен пољем *record* класе *CurrentRecord*. У случају да запис није валидан

формира се извештај у облику *String*-а који садржи опис разлога због којих запис није валидан. Тај извештај се прослеђује као излазни параметар операције *validateRecord()*.

У случају да запис није валидан из операције *saveCurrentRecord()* прослеђује се изузетак са поруком која садржи извештај о невалидности записа. Прекида се рад ове операције и запис неће бити сачуван.

Уколико је запис валидан врши се снимање записа. Поступак снимања записа разликује се у случају да се запис снима први пут или тај запис већ постоји и врши се његово ажурирање. Информацију о томе да ли се запис снима први пут, одређује вредност логичких поља *update* и *savedOnce* које обе имају вредност *false* при првом снимању записа. Уколико се запис снима први пут потребно му је доделити одговарајући *id*, име библиотекара који је формирао запис, тип публикације (добија на основу типа обраде који се користи у едитору), као и датум креирања записа. Након додељивања наведених вредности запис се складишти у базу података и логичка променљива *savedOnce* поставља се на *true*. У случају да се врши снимање записа који је раније формиран и у едитору се само модификује, потребно је поставити датум последње измене записа и ажурирати запис у бази.

4.4.2 Имплементација класе *RecordEditing*

Класа *RecordEditing* садржи операције за обраду самог записа, односно додавање или уклањања елемената записа као и промену садржаја записа. Све операције ове класе обрађују поље *record* класе *CurrentRecord*. Након позива било које операције из ове класе врши се освежавање приказа стабла записа које се формира на основу садржаја објекта *record*. Операције ове класе су:

- *addField(YField)* – додавање поља у запис,
- *addSubfield(YSubfield, String)* – додавање потпоља у запис,
- *addIndicator(YIndicator, String)* – додавање индикатора,
- *deleteField(Field)* – брисање поља,
- *deleteSubfield(Field, Subfield)* – брисање потпоља и
- *changeSubfieldContent(Subfield, String)* – промена садржаја потпоља.

Додавање поља у запис имплементирано је на основу дијаграма активности који је дат у поглављу 3.3.1.

Позивом операције *addSubfield(YSubfield, String)* у запис се додаје потпоље чија спецификација у формату је представљена објектом који се прослеђује као први параматер. Други параметар представља садржај

потпоља које се додаје и који ће у позиву операције бити или празан стринг или подразумевана вредност за потпоље. Потпоље се додаје у поље које је селектовано у стаблу записа и које се налази као садржај атрибута *selectedField* класе *CurrentRecord*. Приликом додавања потпоља у поље најпре се проверава да ли постоји селектовано поље а затим и да ли назив селектованог поља одговара називу поља које је према формату дефинисано као поље које садржи посматрано потпоље. Уколико су ови услови испуњени проверава се да ли би додавање потпоља нарушило валидност записа. То потразумева да треба проверити да ли потпоље са истим именом већ постоји у пољу, уколико постоји проверава се да ли потпоље има особину поновљивости. Уколико потпоље није поновљиво, а потпоље са истим именом већ постоји у пољу прослеђује се изузетак са одговарајућом поруком и потпоље се неће додати. У случају да се установи да додавање потпоља не нарушава валидност записа потпоље се додаје позивом операције *addOnRightPlace(Field, Subfield)* која у пољу које се прослеђује као први параметар проналази одговарајуће место за потпоље (други параметар операције) и додаје га на то место. Одговарајуће место аз потпоље дефинисано је условом да су потпоља у пољу поређана према ознаци у растућем редоследу.

Контроле које се врше приликом промене садржаја потпоља имплементиране су на основу дијаграма активности који је дат у поглављу 3.3.2.

4.4.3 Имплементација класе *RecordFactory*

Класа *RecordFactory* садржи операције за серијализацију и десеријализацију објектног модела библиографског записа.

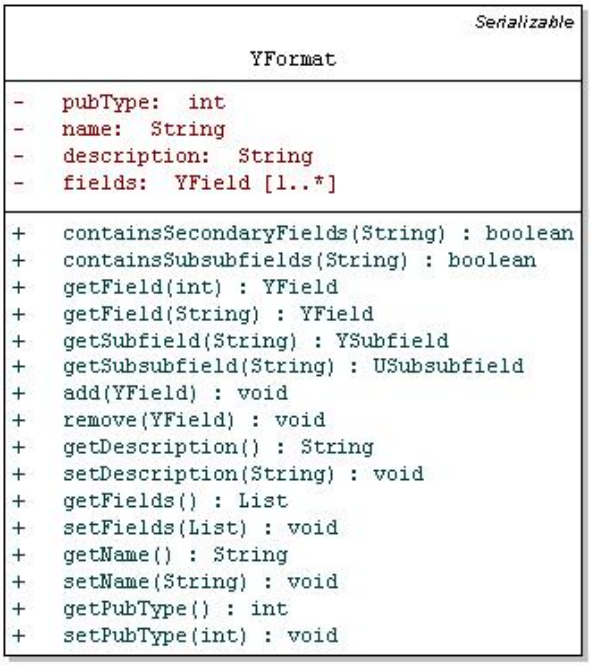
Објектни модел се може представити у облику XML стринга за шта се користе операције *toLooseXML(Record)* и *toTightXML(Record)* које објекат који се прослеђује као параметар серијализују у два различита начина репрезентације записа у облику XML докумената. Операција *toLooseXML(Record)* серијализује запис у XML документ чија спецификација је дата у одељку 2.4. Са друге стране, подаци о запису се могу учитати из XML докумената у објектни модел, зашта се користе операције *fromLooseXML(String)* и *fromTightXML(String)* које парсирају садржај који им се прослеђује као параметар и враћају објекат класе *Record* са учитаним садржајем записа.

Операција *toFullFormat(Record)* враћа *String* који садржи запис у тзв. *full* формату који представља погодан начин за приказ записа. Овај формат користи се и за приказ записа у едитору.

4.5 Објектни модел библиотечког формата

Основна улога меморијске репрезентације библиотечког формата, тачније његовог подскупа јесте приказ података који се односе на сам формат по ком се врши обрада библиграфске грађе.

Приликом покретања едитора на основу окружења пријављеног библиотекара и структуре записа који се учитава у едитор издваја се подскуп библиотечког формата. У току обраде могуће је извршити модификацију издвојеног подскупа. Тај подскуп се у имплементацији едитора чува као инстанца објектног модела библиотечког формата чији дијаграм класа је дат на слици 3.4 и све операције се извршавају над овом инстанцом.

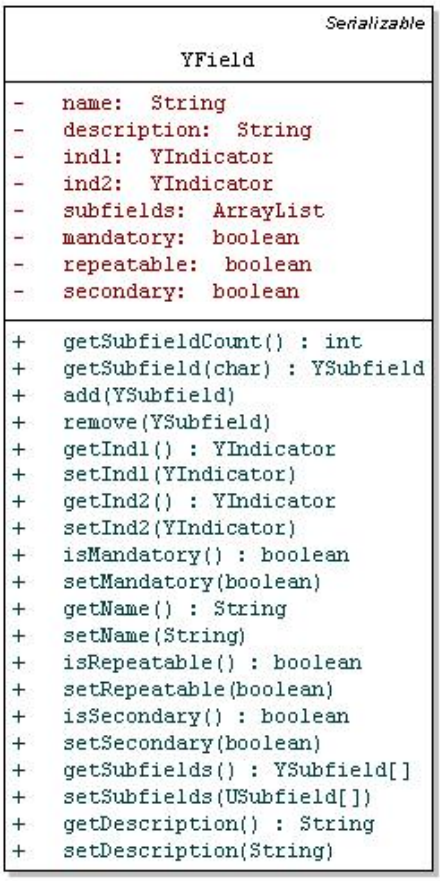


Слика 4.3 *Класа YFormat*

Класе објектног модела библиотечког формата садрже операције за промену садржаја и структуре подскупа библиотечког формата. Поред стандардних *set* и *get* метода за приступ садржајима атрибута, класе овог објектног модела садрже и операције над колекцијама података као што су додавање елемената у листу, уклањање елемената из листе, сортирање елемената листе и др.

На слици 4.3 приказана је класа *YFormat* са свим својим операцијама. За додавања новог поља у објектни модел формата користи се операције *add(YField)*. Операција *getField(String)* враћа поље формата чији назив одговара називу који се прослеђује као параметар

На слици 4.4 приказана је класа која представља поље формата са свим својим операцијама. У скуп операција ове класе спадају *set* и *get* методе за приступ атрибутима класе као и операције за обраду листе потпоља.



Слике 4.4 *Класа YField*

Приликом креирања објектног модела библиотечког формата подаци о формату учитавају се из XML документа који је представљен у поглављу 2. Ово учитавање података врши се уз помоћ операција објектног модела. На пример приликом креирања објекта које представља поље формата операцијама *setName(String)*, *setDescription(String)*, *setMandatory(boolean)*, *setRepeatable(boolean)*, *setSecondary(boolean)*, постављају се редом назив поља, опис поља, и особине обавезности,

поновљивости и секундарности. Операције објектног модела библиотечког формата користе се и приликом контроле валидности записа. Уколико треба проверити да ли неко поље има особину поновљивости позива се његова операција *isRepeatable()* која враћа логичку вредност *true* уколико поље јесте поновљиво.

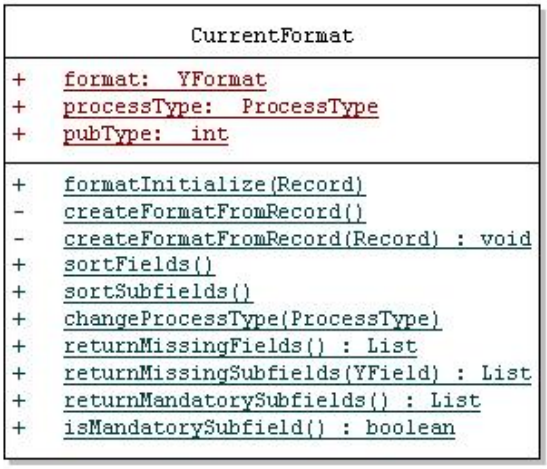
4.6 Имплементација обраде меморијске структуре библиотечког формата

У одељку 3.2.5 описан је статички модел обраде меморијске структуре библиотечког формата по ком се врши обрада библиографске грађе у едитору. Целокупна функционалност едитора која се односи на обраду и коришћење података о формату имплементирана је у класама пакета *FormatEnvironment*.

4.6.1 Имплементација класе *CurrentFormat*

На слици 4.5 приказана је класа *CurrentFormat* са свим својим атрибутима и операцијама.

У класи *CurrentFormat* у атрибуту *format* који је инстанца класе *YFormat* чува се тренутно стање подскупа библиотечког формата који је кориснику приказан у едитору. Све промене које корисник у току рада изврши над стаблом библиотечког формата врше се над овим објектом. Такође, на основу стања овог објекта ажурира се приказ формата у стаблу.



Слика 4.5 *Класа CurrentFormat*

Поред објекта *format* ова класа садржи и атрибуте која садрже податке о типу обраде по ком се тренутно врши обрада у едитору (атрибут *processType*) и типу публикације која се обрађује (атрибут *pubType*).

Операција *formatInitialize(Record)* класе *CurrentFormat* врши иницијализацију поља *format*. Подскуп формата се иницијализује на основу типа обраде пријављеног библиотекара али и записа који је учитан у едитор. Уколико се едитор покреће са учитаним записом објектни модел тог записа се прослеђује као параметар ове операције. Ако се едитор отвара за креирање новог записа као параметар ове операције прослеђује се *null* вредност и у том случају се поље *format* иницијализује само на основу типа обраде. Ова операција је реализована тако што се позивају две помоћне операције, *createFormatFromRecord(Record)* која у објекат *format* додаје елементе из записа и *createFormatFromProcessType()* која у овај објекат додаје елементе који су дефинисани типом обраде.

Класа *CurrentFormat* садржи и операције за модификацију објекта *format* као што су сортирање поља и потпоља, а позивом операције *changeProcessType(ProcessType)* могуће је и променити тип обраде при чему се поље *format* поново иницијализује.

Поред операција за обраду објектног модела формата ова класа садржи и операције које се позивају из едитора у случају да треба додати ново поље или потпоље у запис. Операције *returnMissingFields()* враћа листу објеката класе *YField* који представљају поља која су дефинисана форматом али нису обухваћена подскупом по ком се тренутно обрађује. Ова операција се позива када корисник жели да дода ново поље у стабло формата. Поред тога, операција *returnMissingSubfields (YField)* враћа листу објеката класе *YSubfield* који представљају потпоља поља које се прослеђује као параметар и то она потпоља која су дефинисана у формату али се у тренутној инстани подскупа формата у њему не налазе.

Класа *CurrentFormat* садржи операције које се користе приликом контроле валидности записа. Наиме, приликом обраде записа према неком типу обраде контролише се усклађеност записа са тим типом обраде. Ова контрола односи се на унос потпоља која су у типу обраде дефинисана као обавезна. Потпоља која су по типу обраде дефинисана као обавезна по стандарду не морају имати особину обавезности. Операције *returnMandatorySubfields()* враћа листу обавезних потпоља за актуелни тип обраде. Операција *isMandatorySubfield(String)* враћа логичку вредност *true* уколико је потпоље чији се пуни назив (назив поља иза кога следи назив потпоља, на пример 200a) прослеђује као улазни параметар обавезно по типу обраде, односно *false* уколико није обавезно.

4.6.2 Имплементација класе *FormatFactory*

Класа *FormatFactory* садржи операције за учитавање података о целом библиотечком формату у меморијску структуру. Постоје три операције којима је имплементирано креирање објекта *YFormat*. Свака операција користи различити улазни параметар као извор за податке о формату:

- *getFormat(Connection)* – објектни модел формата добија се из релационе базе података; параметар операције је објекат који представља конекцију на базу, класа *Connection* увози се из Јавине библиотеке *java.sql* [JavaSQL],
- *getFormat(String)* – као улазна информација прослеђује се стринг који садржи податке о формату у облику XML документа чија спецификација је дата у одељку 2.1.1. и
- *getFormat(InputStream)* – као улазна информација прослеђује се улазни ток карактера који такође садржи податке о формату у облику XML документа; класа *InputStream* увози се из *Java*-ине библиотеке *java.io* [JavaIO].

4.6.3 Имплементација класе *PubTypes*

Класа *PubTypes* омогућава добијање података о спецификацији типова публикације. Подаци о типовима публикације иницијализују се при првом позиву било које операције из ове класе јер је та иницијализација имплементирана у *static* блоку класе. Подаци о елементима који припадају типу публикације учитавају се из посебних XML докумената. На основу тих података формира се хеш мапа, односно инстанца класе *HashMap* која се увози из пакета *java.util* [JavaUTIL]. Сваки елемент хеш мапе састоји се од кључа који представља нумеричку ознаку за тип публикације (на пример број 1 је ознака за монографске публикације, а број 2 за серијске) и појаве класе *YFormat* која представља спецификацију посматраног типа публикације, односно садржи сва поља и потпоља која су дефинисана по том типу публикације.

Ова класа садржи следеће операције:

- *getPubTypeCount()* – враћа број дефинисаних типова публикације, односну величину хеш мапе и
- *getPubType(int)* – враћа спецификацију типа публикације чија нумеричка ознака се прослеђује као параметар.

4.3 Графичка компонента за репрезентацију стабла – *JTree*

Графичка компонента која је коришћена приликом имплементације основне функционалности која се односи на обраду структуре како

библиографског записа тако и библиотечког формата је компонента *JTree* [JTree04] која представља стандардну компоненту за графичку репрезентацију стабла платформе *Swing*. Поред класе *JTree* у оквиру *Swing* платформе реализован је велики број класа за манипулацију са компонентом за репрезентацију стабала. Ове класе се могу наћи у пакету *javax.swing.tree*.

Компонента *JTree* омогућава графички приказ хијерархијских података. Ова компонента је реализована тако да објекат који је инстанца класе *JTree* не садржи податке, већ само обезбеђује њихов приказ на екрану.

Као и свака сложенија *Swing* компонента, стабло чита податке из модела података (*data model*) који се добија имплементацијом интерфејса *TreeModel*. Спецификација модела података за стабло добија се имплементацијом операција којима се описује структура стабла. Ту спадају операције као што су спецификација коренског објекта, одређивање потомка чвора на основу редног броја, одређивање броја потомака чвора, да ли је одређени чвор у стаблу лист (нема потомака) и слично. У моделу података стабла чвор стабла може бити било који објекат.

Приликом промене података у стаблу, тачније у моделу података на основу ког се креира стабло, освежавање приказа постиже се позивом операције *fireTreeStructureChanged()* која је дефинисана у моделу података.

За спецификацију изгледа чвора користи се интерфејс *TreeCellRenderer*. Имплементацијом овог интерфејса дефинише се репрезентација текста објеката који се јављају као чворови стабла, изглед текста (боја, величина и тип слова, боја позадине селектованог и неселектованог чвора), као и иконица чвора. Имплементацијом поменутог интерфејса могуће је дефинисати различит начин приказа чвора у стаблу у зависности од тога да ли је чвор отворен или затворен, да ли је селектован, да ли има фокус, да ли је чвор лист у стаблу и на који се објекат чвор односи.

Компонента *JTree* подржава директан унос података у стабло. Специфицификација компоненте за едитовање чвора стабла постиже се имплементацијом интерфејса *TreeCellEditor*.

Сви поменути интерфејси и класе за спецификацију изгледа и функционалност стабла налазе се у пакету *javax.swing.tree*.

4.3 Имплементација стабла библиотечког формата

Стабло библиотетечког формата у XML едитору за обраду библиографске грађе служи за приказ библиотечког формата или подскупа формата по

ком се врши обрада библиографске грађе и селекцију елемената формата ради уноса у библиографски запис. Ово стабло садржи податке о елементима формата који се у стаблу јављају као чворови.

Модел података за стабло формата креира се класом *FormatTreeModel* која имплементира интерфејс *TreeModel*. Модел се креира на основу објектног модела YUMARC формата чији дијаграм класа је описан у одељку 3.2.3. Овај објектни модел се приликом иницијализације стабла учитава из XML документа формата. Као чворови стабла јављају се објекти из објектног модела формата, а као коренски елемент наводи си назив подскупа формата по ком се врши обрада библиографске грађе.

Класом *FormatTreeCellRenderer* која имплементира интерфејс *TreeCellRenderer* дефинисан је приказ елемената формата у стаблу. У приказу елемената формата наводи се његов назив и опис као и информација која се односи на особину поновљивости елемента.

У едитору су имплементиране операције за модификацију стабла формата. На пример, могуће је у стабло формата додати поље или потпоље формата. Све операције које модификују стабло формата врше се над објектним моделом који специфицира модел података за стабло формата, при чему се након извршене промене освежава приказ стабла позивом операције *fireTreeStructureChanged()*.

4.4 Имплементација стабла библиографског записа

Стабло библиографског записа заузима централно место у процесу креирања библиографског записа. Ова структура служи за приказ библиографског записа, а над њом се врши и обрада библиографске грађе.

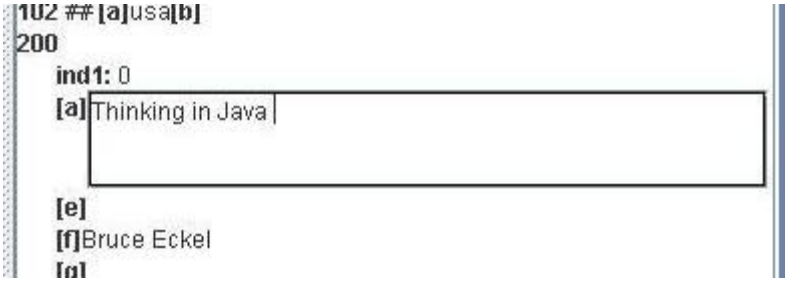
Модел података за стабло формата креира се класом *RecordTreeModel* који имплементира интерфејс *TreeModel*. Модел података се креира на основу објектног модела библиографског записа по YUMARC формату чији дијаграм класа је дат у одељку 3.2.2 Као чворови стабла библиографског записа јављају се елементи библиографског записа у *full* формату.

Класом *RecordTreeCellRenderer* која имплементира интерфејс *TreeCellRenderer* дефинасан је изглед елемената записа у стаблу. Изглед поља записа дефинисан је различито у зависности од тога да ли је тај чвор у стаблу отворен или затворен. На слици 4.2 приказана су два различита начина приказа чвора који представља поље *200* у стаблу записа. На слици означеној са (а) приказан је изглед затвореног чвора поља *200*, док слика означена са (б) представља изглед отвореног чвора поља *200*. Ради боље прегледности називи елемената записа су истакнути у стаблу.

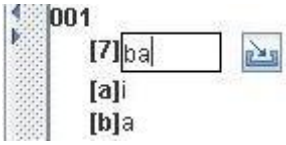
(а) (б)

Слика 4.2 *Два начина приказа поља 200 у стаблу библиографског записа*

Једна од главних улога стабла библиографског записа је унос библиографских података. Да би подаци могли да се уносе директно у стабло, тачније у чворове стабла потребно је за стабло позвати метод *setEditable(true)* и дефинисати текстуални едитор за унос података имплементацијом интерфејса *TreeCellEditor*. Текстуални едитор за унос података у чворове стабла дефинисан је класом *RecordTreeCellEditor* која имплементира интерфејс *TreeCellEditor*. У овој класи је дефинисано који чворови могу да се едитирају, а то су само они чворови који немају потомке у стаблу библиографског записа, односно индикатори и потпоља. Текстуални едитор за унос библиографских података у чворове стабла имплементиран је различито у зависности од тога да ли се уноси произвољан текст или шифрирана вреднот. На слици 4.3а приказан је едитор за унос слободног садржаја у чвор који представља потпоље *а* поља *200*. На слици 4.3б приказан је начин имплементације уноса шифриране вредности у потпоље *7* поља *001*. Дугме са иконицом  које је приказано на слици 4.3б има функционалност отварања шифарника.



Слика 4.3а *Текстуални едиотр за унос вредности потпоља а поља 200*



Слика 4.3б ***Текстуални едитор за унос шифриране вредности у потпоље 7 поља 00***

Све промене које се у стаблу библиографског записа изврше над библиографским записом, било да се ради о промени структуре или промени садржаја, врше се над објектним моделом записа који служи као основа за креирање модела стабла записа. Након извршене промене података у објектном моделу освежава се приказ стабла библиографског записа позивом операције *fireTreeStructureChanged()*.

Приликом обраде библиографског записа врши се контрола валидности записа. Тако на пример, пре прихватања вредности која се унесе као садржај неких од описаних текстуалних едитора за унос садржаја чворова у стаблу врше се провере које су описане дијаграмом активности у одељку 3.3.2.

Поглавље 5

Опис коришћења едитора за обраду библиографске грађе

XML едитор за обраду библиографске грађе реализован је као независна софтверска компонента. Верификација и тестирање едитора извршени су његовом интеграцијом у четврту верзију библиотечког софтверског система БИСИС.

Поред обраде библиографске грађе, систем БИСИС обухвата и додатне функционалности у области библиотекарства. Ту спадају складиштење библиографских записа, подешавање окружења библиотекара, дефинисање улога и овлашћења за кориснике, претраживање библиографских записа, креирање разних врста извештаја, креирање каталошких листића [Rađenović], обрада локацијских података и коришћење библиотечке грађе у раду са члановима библиотеке [Tešendić].

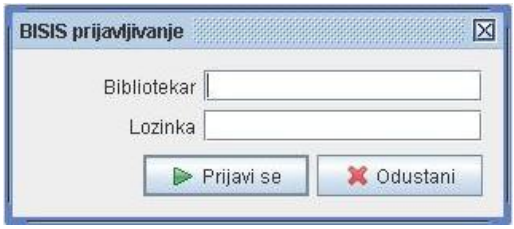
У овом поглављу дат је опис коришћења едитора за обраду библиографске грађе у оквиру четврте верзије библиотечког софтверског система БИСИС. Описане су следеће функционалности овог система:

- претраживање базе библиографских записа,
- креирање библиографског записа у едитору,
- обрада локацијских података.

Опис коришћења едитора илустрован је над базом библиографских записа библиотеке Департмана за математику и информатику Природно-математичког факултета у Новом Саду.

5.1 Пријављивање на систем

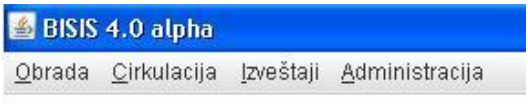
Након стартовања апликације добија се прозор за пријављивање библиотекара, приказан на слици 5.1. Потребно је у одговарајућа поља унети корисничко име и лозинку при чему се мала и велика слова не разликују. У пољу за унос лозинке унети знаци замењују се звездицама (*).



Слика 5.1 *Пријављивање библиотекара*

Притиском на дугме *Prijavi se* на основу улога дефинисаних за библиотекара који се пријављује, формира се главни мени апликације. Библиотекар може имати дефинисане следеће улоге: администрација, обрада и циркулација.

На слици 5.2а приказан је главни мени апликације у случају да пријављени библиотекар има дефинисане све три улоге (администрација, обрада и циркулација). На слици 5.2б приказан је главни мени апликације у случају да пријављени библиотекар има дефинисану само улогу за обраду. Слова у називу менија која су подвучена представљају пречицу за отварање менија. Њиховом комбинацијом са тастером <Alt> могуће је коришћењем тастатуре отворити одговарајући мени.



Слика 5.2а *Главни мени апликације*

Слика 5.2б *Главни мени апликације*

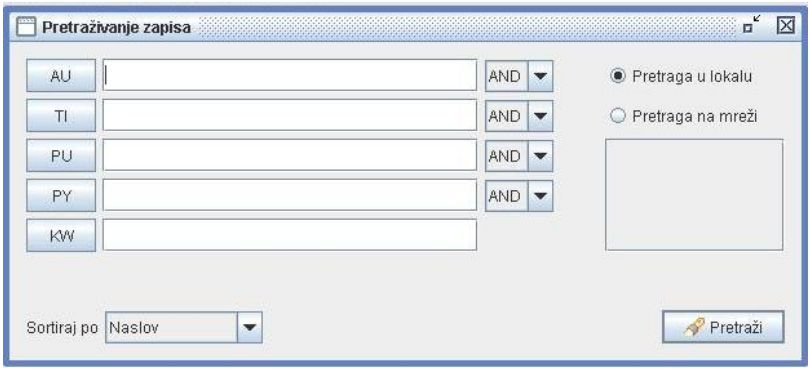
Мени *Obrada* садржи акције које се односе на обраду библиографске грађе и обухватају коришћење едитора за обраду библиграфске грађе. На слици 5.3 приказане су акције менија *Obrada*. Акцијом *Pretraživanje* отвара се прозор за претраживање библиографске грађе, а акцијом *Novi zapis...* отвара се едитор за обраду библиографске грађе и то са празним записом. Поред назива акције наведен је назив тастера који представља пречицу до ове акције. Без обзира да ли је мени апликације отворен притиском на тастер чији назив је наведен поред назива акције покреће се одговарајућа акција.

Слика 5.3 *Подменији менија Obrada*

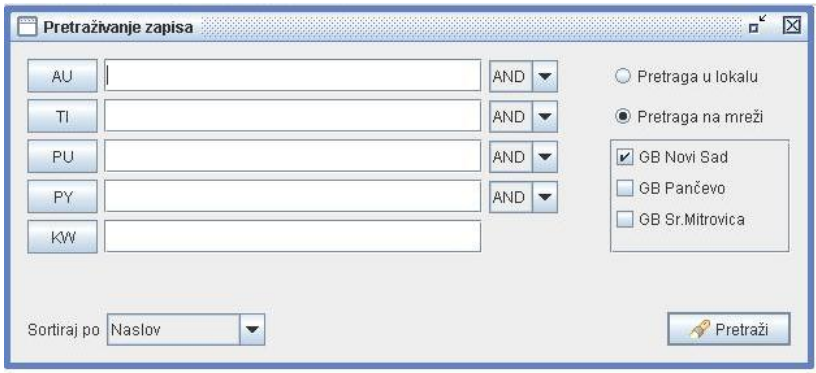
Скуп акција које ће садржати мени обрада такође зависи од улоге пријављеног библиотекара. Сви библиотекари, без обзира на улогу, могу да претражују базу записа, тако да ће се акција *Претраживање* бити доступна свим библиотекарима, док ће акција *Novi zapis...* бити доступна само ако пријављени библиотекар има дефинисану улогу за обраду.

5.2 Претраживање библиографске грађе

Прозор за претраживање библиографске грађе приказан је на слици 5.4. При отварању ове екранске форме подразумева се претраживање у локалу (у горњем десном углу селектовано *Pretraga u lokalu*). На овој екранској форми може се изабрати претраживање библиографских записа у библиотекама које припадају БИСИС заједници. На слици 5.5 приказан је изглед прозора за претраживање приликом претраге мреже библиотека (у горњем десном углу селектовано *Pretraga u mreži*). Овим селектовањем добија се листа за избор удаљених библиотека које припадају библиотечкој мрежи БИСИС.



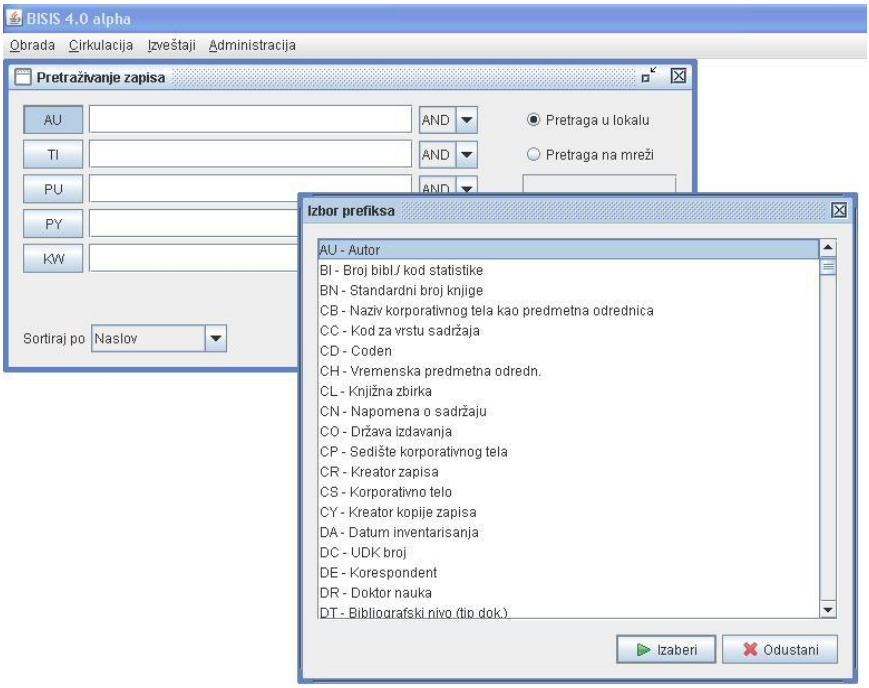
Слика 5.4 *Прозор за претраживање у локалу*



Слика 5.5 *Претрага у библиотечкој мрежи БИСИС*

На прозорима за претраживање (слика 5.4 и 5.5) налази се пет линија за претраживање, падајућа листа за избор критеријума за сортирање резултата претраге и компонента за избор локације за претрагу.

Свака од линија за претраживање састоји се од префикса по ком се врши претрага и који се може изабрати из шифарника префикса, једнолинијског едитора за унос појмова за претраживање, а прве четири и од падајуће листе за избор оператора. Притиском на дугме чија лабела је назив префикса или на тастер <F9> отвара се прозор за избор префикса приказан на слици 5.6.



Слика 5.6 *Избор префикса за претраживање*

Приликом претраживања могуће је дефинисати највише пет операнада са логичким операторима који их повезују. Операнд се састоји од префикса и појма за претраживање. Уколико библиотекар не унесе никакав појам, програм неће обавити претраживање по датом префиксу.

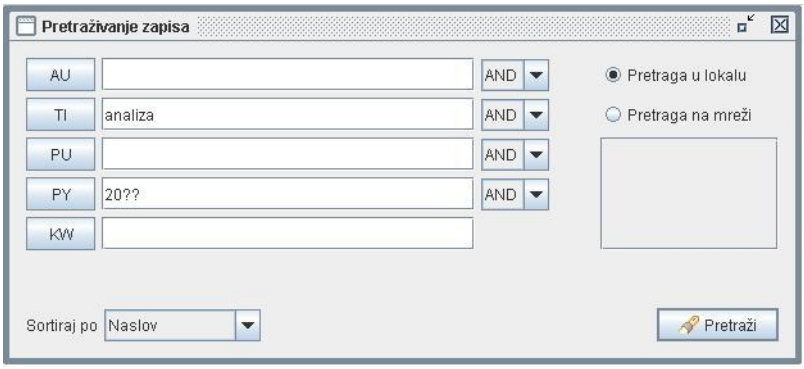
Приликом претраживања не разликују се велика и мала слова, као и латинично и ћирилично писмо, чиме је омогућено да се без обзира у ком формату (велика и мала слова) или ком писму је унет садржај у једнолинијски едитор пронађу сви записи који садрже тражену реч или фразу. Такође, претрага не зависи ни од формата и писма у ком је та реч или фраза унета у запис.

Веза између префикса су оператори који се бирају из падајуће листе. Постоје три могућа оператора AND, OR и NOT.

У појмовима за претраживање може се наћи и специјалан знак тилда (~) и то као први или последњи знак. Уносом тилде испред речи која се тражи претражује се садржај префикса који почиње том речју. Слично, тилда на крају речи која се претражује омогућава претраживање садржаја префикса који се завршева том речју.

Поред тога, у појмовима за претраживање могу се налазити и специјални знаци “*” и “?”. Знак “*” замењује нула или више знакова, а знак “?” највише један знак.

На слици 5.7 приказана је екранска форма за претраживање са једним упитом, који гласи: наслов дела треба да садржи реч „analiza“ и да је издата године која почиње са “20” односно после 1999. године.

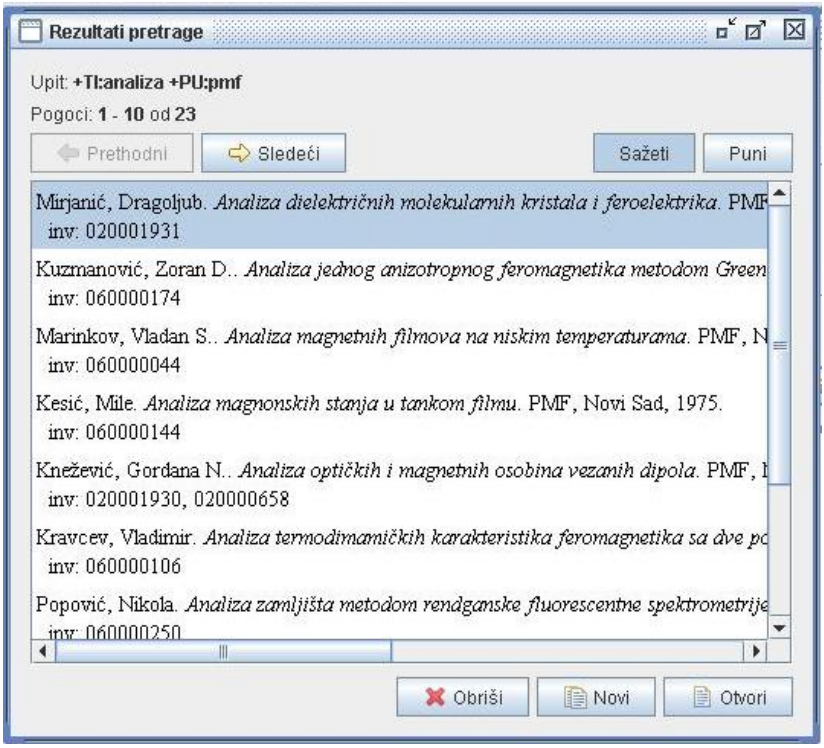


Слика 5.7 *Прозор за претраживање са упитом*

Притиском на дугме *Pretraži* извршава се претрага према задатим параметрима.

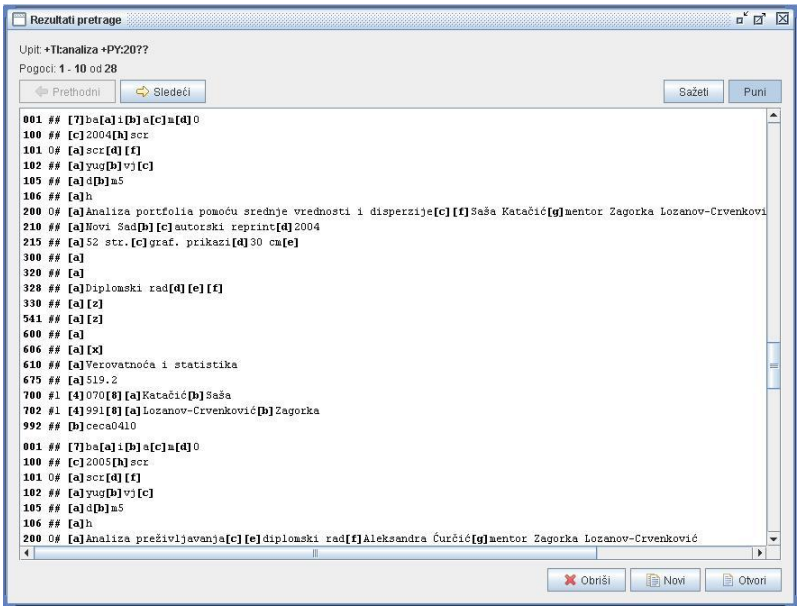
5.2.1 Приказ резултата претраге

Уколико има пронађених погодака приказује се форма за приказ резултата претраге. На слици 5.8а приказан је пример резултата претраге за упит са слике 5.7.



Слика 5.8а *Приказ резултата претраге*

У горњем делу прозора који је приказан на слици 5.8а наведен је упит по ком је извршена претрага. На прозору се налазе дугмад за навигацију по скупу резултата претраге (*Prethodni* и *Sledeći*), као и дугмад за избор начина приказа резултата (*Sažeti* и *Puni*). Највећи део прозора за приказ резултата заузима листа погодака (централни део екранске форме) која садржи сажети приказ пронађених записа (слика 5.8а). Притиском на дугме *Puni* (слика 5.8б) пронађени записи се приказују у *full* формату.



Слика 5.8б *Приказ резултата претраге*

У доњем делу прозора за приказ резултата налазе се три дугмета са следећим функционалностима:

- *Obriši* – брисање селектованог записа,
- *Novi* – креирање новог записа на основу селектованог записа и
- *Otvori* – модификација селектованог записа.

Покретањем једне од последње две функције селектовани запис се учитава у едитор и приказује се главна екранска форма едитора.

5.3 Обрада библиографског записа у едитору

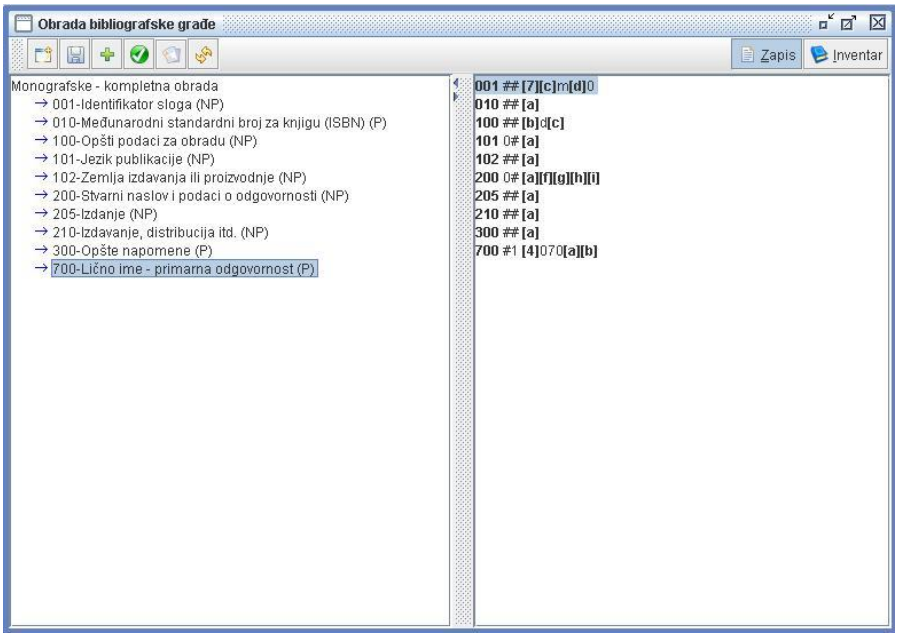
Едитор за обраду библиографске грађе у систему БИСИС могуће је отворити на три начина. Први је позивом акције *Novi zapis...* из подменија менија *Obrada* (слика 5.3). На овај начин отвара се нови запис у едитору који је приказан на слици 5.9. Друга два начина за покретање едитора представљају отварање већ постојећег записа у едитору и то помоћу акција из прозора за приказ резултата претраге (слика 5.8).

Приликом покретања едитора као улазни параметар користи се окружење библиотекара који је пријављен на систем. Сваки библиотекар који има дефинисану улогу за обраду има дефинисане типове обраде од којих је један означен као подразумевани (*default*) тип обраде и он ће се користити уколико библиотекар не изабере неки други тип обарде. Један тип обраде дефинише се на основу типа публикаце на који се односи,

скупа иницијалних потпоља и скупа обавезних потпоља. Да би запис био валидан у односу на одређени тип обраде морају бити унета сва потпоља коју су по дефиницији типа обраде обавезна. Избор типа обраде описан је у одељку 5.3.2.

5.3.1 Отварање новог записа у едитору

Приликом отварања новог записа у едитору у стаблу формата (лева страна екранске форме) и стаблу записа (десна страна екранске форме) налази се одговарајући скуп поља и потпоља (слика 5.9). Који ће се скуп поља и потпоља појавити зависи од подразумеваног типа обраде за пријављеног библиотекара. Сва потпоља која се налазе у скупу иницијалних потпоља за тај тип обраде налазиће се заједно са својим припадајућим пољима у стаблу формата и стаблу записа. Поред тога, одређена потпоља у запису садрже вредности које су по стандарду за њих дефинисане као подразумеване (*default*). На пример, потпоље *c* поља *001* које представља библиографски ниво публикације као подразумевану вредност има шифру *m* која значи да се ради о монографској публикацији.

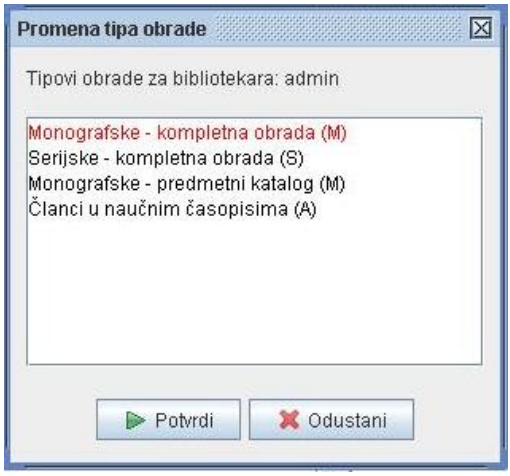


Слика 5.9 *Креирање новог записа у едитору*

5.3.2 Промена типа обраде

Пре уноса било ког податка у запис могуће је променити тип обраде притиском на дугме са иконицом . Притиском на ово дугме добија се прозор приказан на слици 5.10. Оног тренутка када корисник унесе први податак у запис ово дугме престаје да буде активно и мора се наставити

обрада по изабраном типу обарде. Промена типа обраде сада се може извршити тек након напуштања записа чија обрада је у току.



Слика 5.10 *Промена типа обраде*

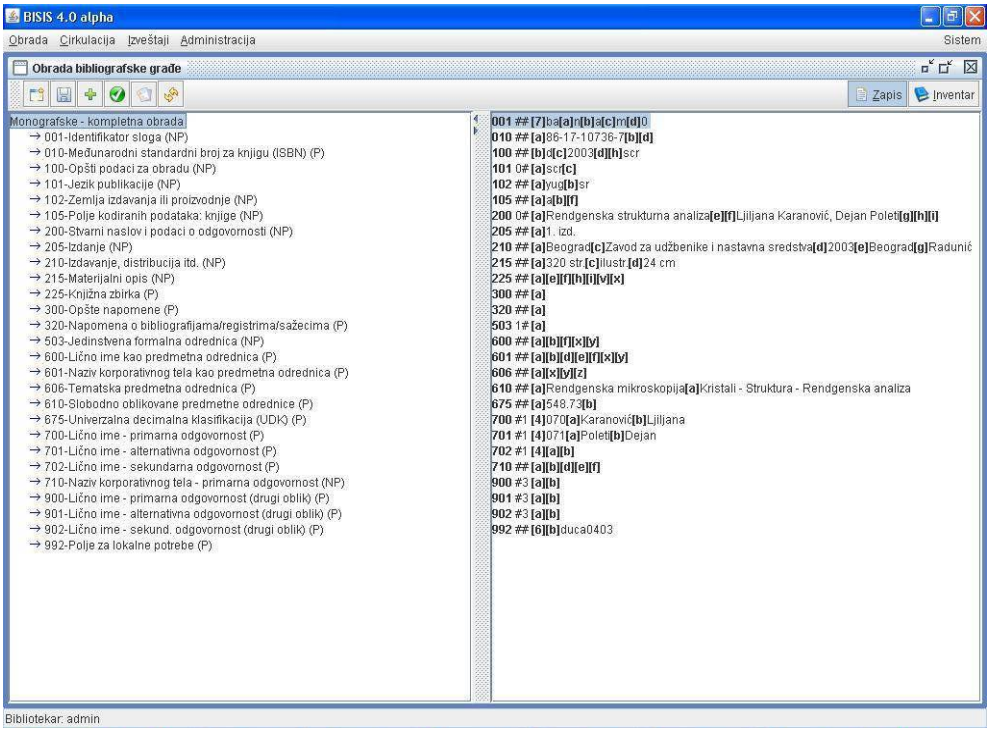
На прозору приказаном на слици 5.10 налази се списак свих типова обраде који су дефинисани за пријављеног библиотекара. За сваки тип обраде наводи се његов назив и у загради почетно слово типа публикације на који се односи. Подразумевани тип обраде за пријављеног библиотекара означен је црвеном бојом. Промена типа обраде постиже се тако што се селектује жељени тип обраде у понуђеној листи и притисне дугме *Potvrdi*.

Промена типа обраде подразумева промену стабла формата и стабла библиографског записа који се сада формирају на основу новоизабраног типа обраде.

5.3.3 Учитавање постојећег записа у едитор

Притиском на дугме *Novi* или *Otvori* на екранској форми за приказ резултата претраге (слика 5.8) селектовани запис се отвара у едитору, и то у првом случају ради креирања новог записа на основу постојећег (селектованог) а у другом ради модификације селектованог записа. Пре отварања записа у едитору проверава се да ли пријављени библиотекар има тип обраде чији тип публикације одговара типу публикације изабраног записа. Уколико нема, то значи да библиотекар није овлашћен за обраду типа публикације на коју се запис односи и пријављује се одговарајућа порука. Ако постоји одговарајући тип обраде, отвара се едитор при чему се сада стабла библиотечког формата и библиографског записа формирају на основу тог типа обраде и структуре самог записа који се отвара. Односно, обе структуре ће садржати поља и потпоља из записа са додатком потпоља која су обухваћена типом обраде.

На слици 5.11 приказан је изглед едитора приликом учитавања једног записа из листе резултата претраге.

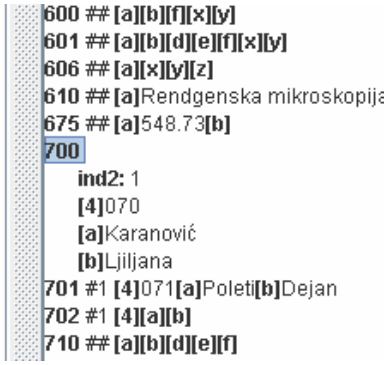


Слика 5.11 *Постојећи запис учитан у едитор*

5.3.4 Обрада записа у едитору

Обрада библиографских записа врши се над стаблом записа, док стабло библиотечког формата служи за селекцију елемената који се додају у запис. Поред тога, основна улога стабла библиотечког формата је да кориснику омогући увид у све информације које се тичу формата (називе поља, потпоља и индикатора, особине поновљивости, и друго).

Стабло библиографског записа приказује структуру записа, његова поља и потпоља као и садржај записа, односно садржај потпоља и индикатора. Кретање по пољима записа врше се притиском на тастере <↑> и <↓>. Притиском на неки број на тастатури селектује се прво поља чији назив почиње тим бројем. Притиском на тастер <Enter> отвара се селектовано поље. На слици 5.12 приказан је пример отвореног поља 700. На овај начин омогућава се кретање по индикаторима и потпољима отвореног поља.



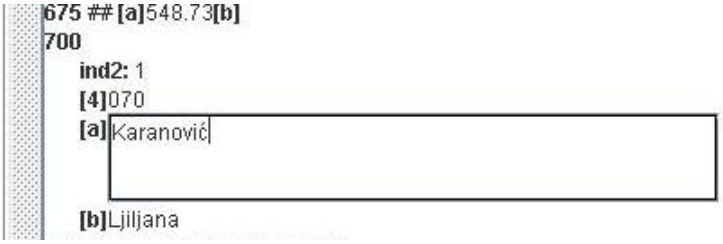
Слика 5.12 *Отворено поље 700*

Над записом се у едитору могу вршити следеће операције:

- промена садржаја записа
- унос садржаја потпоља и индикатора
- промена структуре записа
    - додавање поља и потпоља
    - брисање поља и потпоља
    - промена редоследа потпоља

Промена садржаја записа

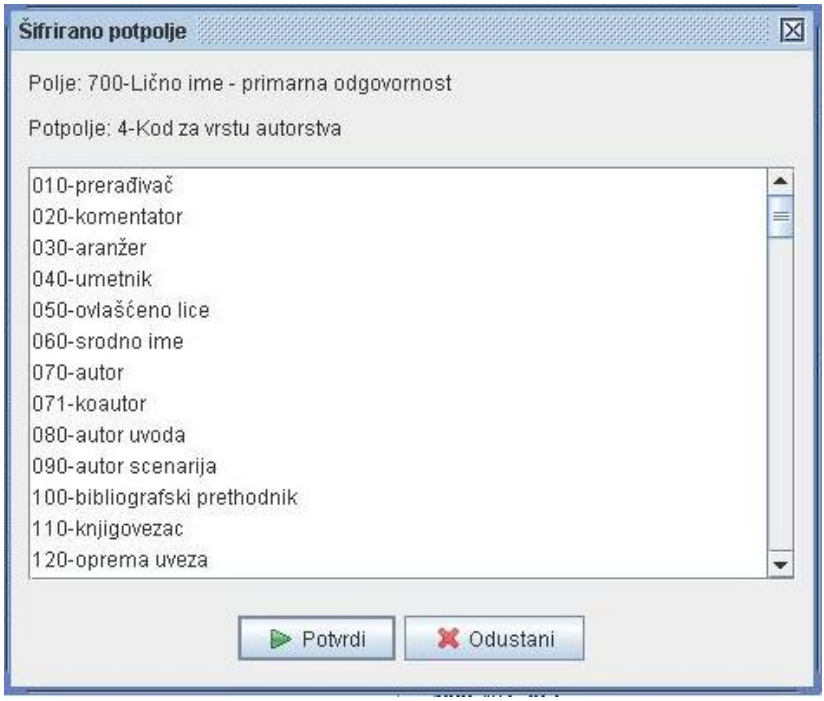
Садржај потпоља или индикатора уноси се у посебне мале едиторе који се отварају позиционирањем на потпоље или индикатор и притиском на тастер <Enter>. Разликују се две врсте едитора за унос садржаја. На слици 5.13а приказан је едитор за унос садржаја нешифрираног потпоља, а на слици 5.13б едитор за унос садржаја шифрираног потпоља. Едитор за унос вредности за индикаторе одговара едитору који је приказан ан слици 5.13б.



Слика 5.13а *Едитор за унос нешифрираног потпоља*

Слиак 5.13б *Едитор за унос шифриране вредности*

Уколико се ради о уносу шифрираних вредности, корисник може да унесе шифру, али може и да преузме вредност из одговарајућег шифарника. Шифарник се отвара притиском на дугме са иконицом поред текстуалног едитора за унос шифре или притиском на тастер <F9> на тастатури. На слици 5.14 приказан је изглед шифарника за потпоље *4* поља *700*.



Слика 5.14 *Шифарник потпоља 4 поља 700*

Шифра се из шифарника може изабрати стрелицом или мишем. Селектована шифра се копира у шифарник притиском на дугме *Potvrdi*, на типку <Enter> или двокликом миша на одговарајућу шифру.

Вредност индикатора уноси се на исти начин као и вредност шифрираног потпоља.

Након уноса садржаја, било шифрираног или не, његово прихватање се врши притиском на тастер <F12> или <Enter>. Приликом

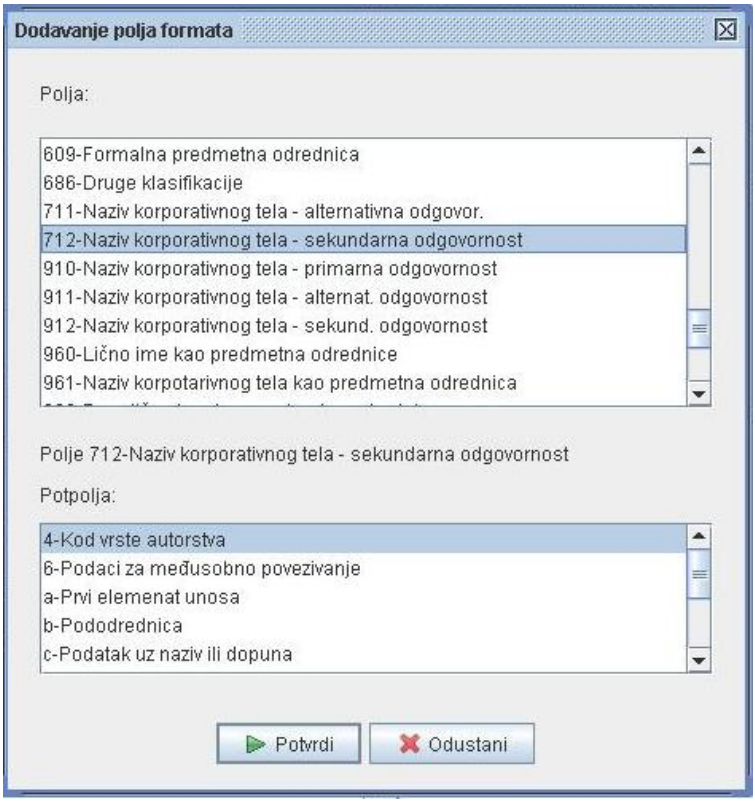
прихватања вредности врши се контрола унетог садржаја. За шифрирана потпоља проверава се исправност шифре, а за нека потпоља проверава се исправност формата у ком је податак унет (ISBN, ISSN број, унос датума и године). У случају да је нарушена валидност записа пријавиће се одговарајућа порука и садржај неће бити прихваћен.

Промена структуре записа

Код **додавања поља и потпоља** разликују се два случаја:

- додавање поља/потпоља које не постоји у запису и у стаблу формата и
- додавања поновљивог поља/потпоља које већ постоји у запису и у стаблу формата.

У првом случају операција додавања поља/потпоља врши се над стаблом формата, али се то поље/потпоље истовремено додаје и у запис.



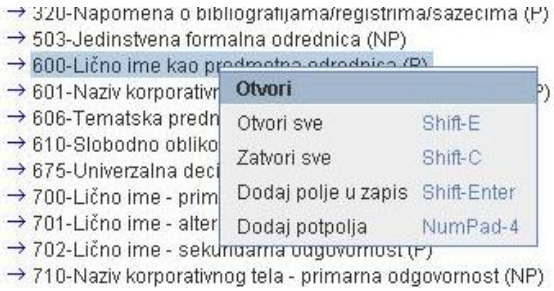
Слика 5.15 *Додавање поља формата*

Поље се у стабло формата може додати притиском на дугме са иконицом ✚ на панелу за брзи приступ командама или пречицом са

тастатуре <NumPad-7> која представља седмицу на нумеричкој тастатури. На слици 5.15 приказан је прозор за додавање поља.

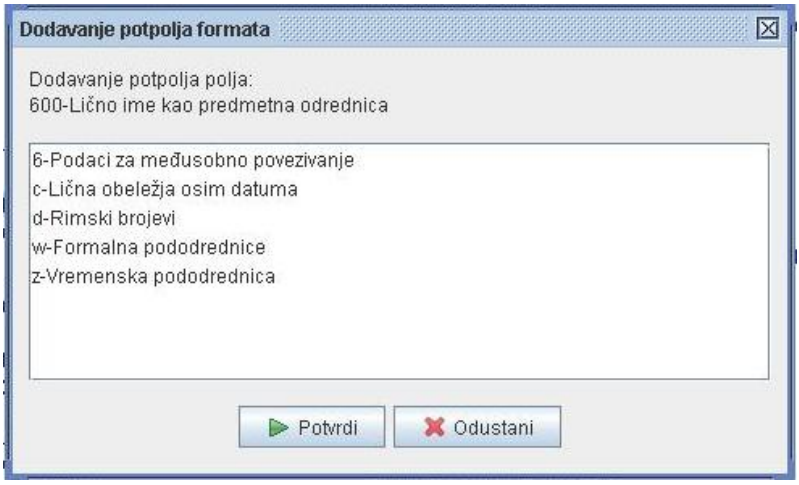
Приликом оваквог додавања поља за свако поље које се додаје мора се додати и макар једно потпоље које се селектује из доње листе. Након селекције одговарајућег поља и његових потпоља притиском на дугме *Potvrdi* или на тастер <Enter> она се додају у стабло формата и у запис.

Додавање потпоља у стабло формата врши се селекцијом поља коме треба додати потпоља и пречицом <NumPad-4> или избором одговарајуће акције (*Dodaj potpolja*) из падајућег менија који се добија десним кликом миша на одговарајуће поље (слика 5.16). Уколико селектовано поље нема додатних потпоља ова акција ће бити светлије обојена и неће бити активна.



Слика 5.16 *Падајућа листа за поље у стаблу формата*

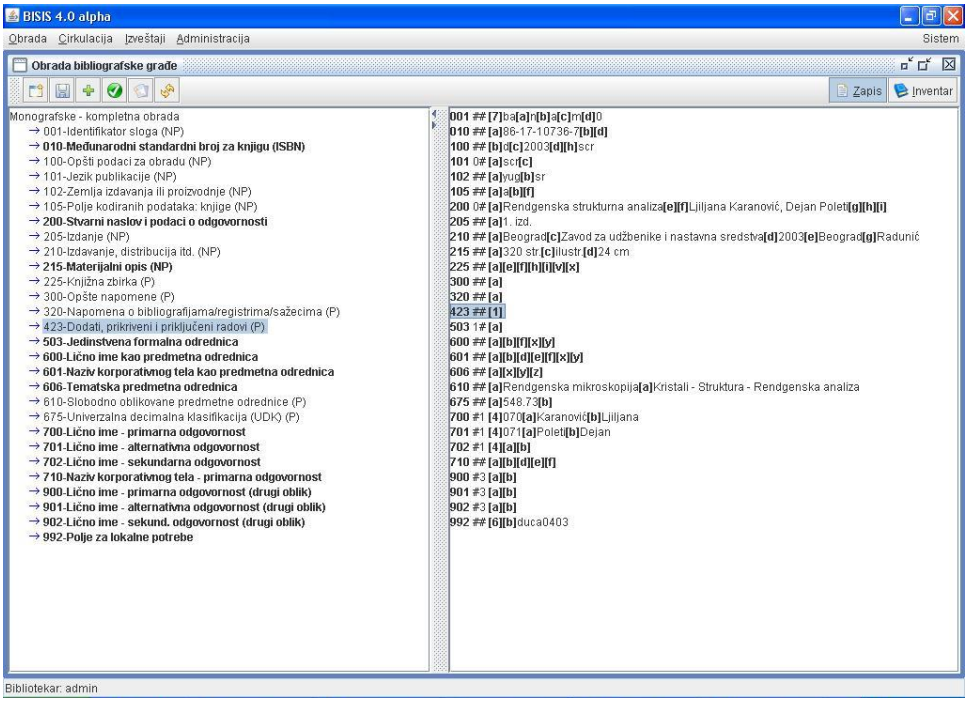
На слици 5.17 приказан је прозор за избор потпоља за додавање. Потпоља се додају тако што се селектују одговарајућа потпоља и притисне се на дугме *Potvrdi* или на типку <Enter>. Изабрана потпоља додају се у стабло формата и у запис.



Слика 5.17 *Додавање потпоља*

Додавање поновљивог поља врши се тако што се поље које треба додати селектије у стаблу формата и позиве се одговарајућа акције. Акција за додавање поља покреће се пречицом <Shift>+<Enter> или акцијом *Dodaj polje u zapis* из падајућег менија који је приказан на слици 5.16. На овај начин у запис ће се додати поље које је селектовано у стаблу формата. Додавање поља у запис подразумева и додавање индикатора са њиховим подразумеваним (*default*) вредностима. Уколико се додаје поље које већ постоји у запису, а нема особину поновљивости пријављује се порука о грешци.

Потпоље се у запис додаје тако што се у стаблу записа прво селектује поље у које треба додати потпоље, затим се потпоље изабере у стаблу формата и двокликом миша или притиском на тастер <Enter> додаје у поље које је селектовано у стаблу записа. Уколико корисник покуша да дода потпоље које није поновљиво, а потпоље са тим називом већ постоји у селектованом пољу пријавиће се одговарајућа порука.

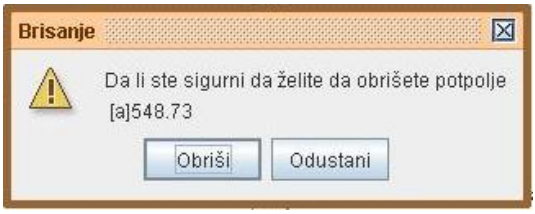


Слика 5.18 *Додавање секундарних поља*

Посебан случај додавања поља представља **додавање секундарних поља** чија је особина да се она јављају као садржај других поља и то поља из блока 4. Ако се у стаблу записа селектује поље које може садржати секундарна поља у стаблу формата се истакну поља која имају особину секундарности. На слици 5.18 приказан је изглед едитора у случају да се у стаблу записа селектује поље *423*. Сада се акцијом за

додавање поља постиже да се та поља додају у поље *423* као секундарна поља и у овом случају могу се додати само поља која имају особину секундарности. Додавање потпоља у секундарна поља врши се исто као и приликом додавања потпоља у обична поља. Селектује се секундарно поље и акцијом за додавање потпоља потпоље се додаје у то селектовано секундарно поље.

Брисање поља и потпоља врши се селекцијом одговарајућег елемента који треба да се обрише у стаблу записа и притиском на тастер <Delete>. Уколико је елемент који се брише обавезан, пријављује се одговарајућа порука и елемент не може бити обрисан. Пре него што се елемент и заиста обрише, кориснику се тражи додатна потврда о брисању (слика 5.19). Тек притиском на дугме *Obriši* елемент записа ће заиста бити обрисан.



Слика 5.19 *Брисање потпоља*

Промена редоследа потпоља у оквиру поља врши се у облику померања селектованог потпоља за једно место испред или иза помоћу тастера <Alt>+<↑> односно <Alt>+<↓>.

5.3.5 Обрада локацијских података

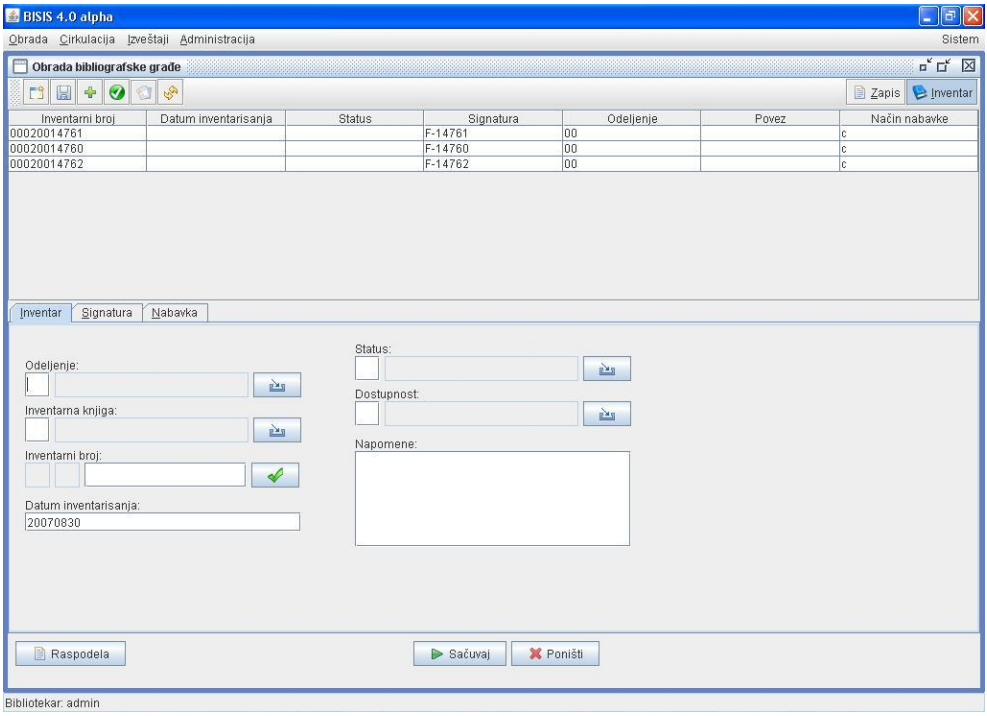
Поред креирања библиографског записа, обрада библиографске грађе обухвата и унос локацијских података, односно података о конкретним примерцима публикације које библиотека поседује. Екранска форма за унос локацијских података за библиографску јединицу која се обрађује у едитору добија се притиском на дугме *Inventar* у горњем десном углу прозора едитора. Изглед екранске форме за унос локацијских података за монографске публикације дат је на слици 5.20.

По YUMARC формату за локацијске податке је резервисано поље *996* у случају монографских публикација, односно поље *997* у случају серијских публикација. Едитор за обраду библиографске грађе реализован је тако да се локацијски подаци уносе независно од библиотечког формата у релациону базу података, па и одговарајућа екранска форма представља стандардну графичку апликацију за унос података у релациону базу података.

Екранска форма за унос локацијских података састоји се из три дела. У горњем делу налази се табела са основним подацима за све

примерке који су унети за пубикацију која се обрађује у едитору. У средњем делу налази се форма за унос података о једном примерку, а у доњем дугмад са акцијама за чување примерака, иницијализацију екранске форме и отварања форма за расподелу примерака.

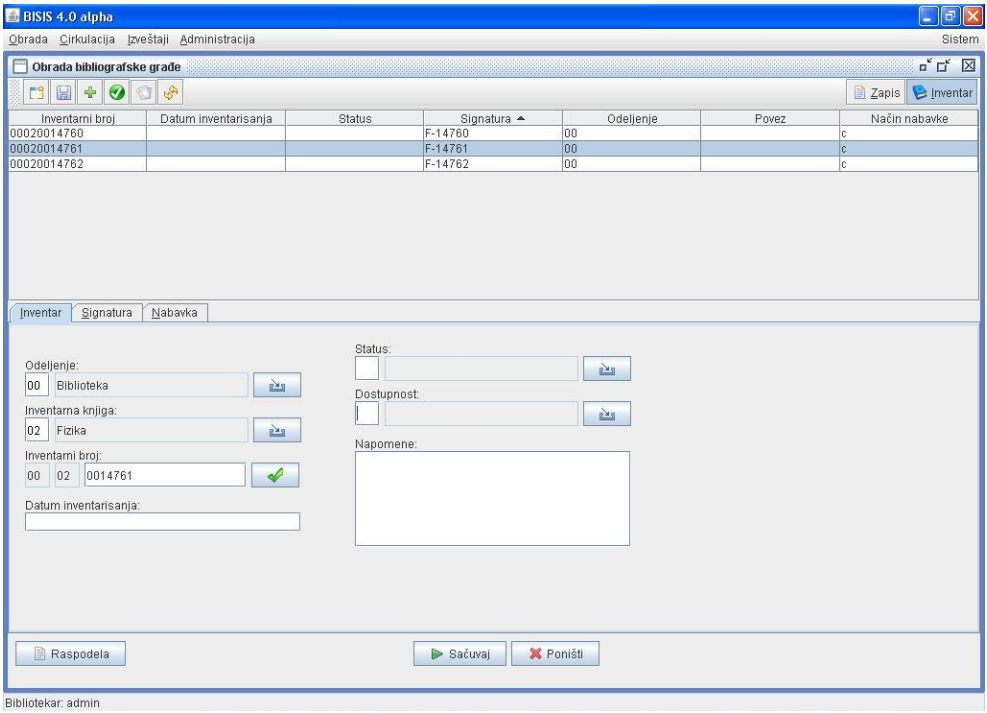
Над табелом за приказ свих примерака може се извршити сортирање по било којој колони и то или у опадајућем или у растућем редоследу. Сортирање по одређеној колони врши се кликом миша на заглавље те колоне. Први клик обезбеђује сортирање у растућем редоследу а други у опадајућем.



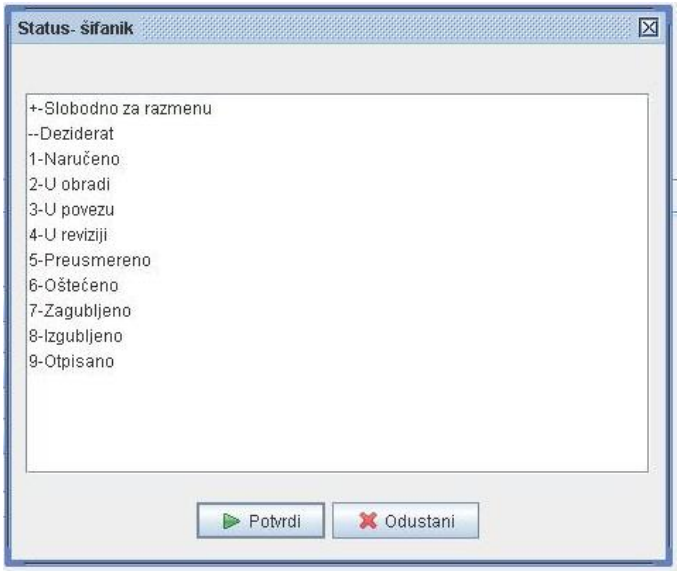
Слика 5.20 *Екранска форма за унос локацијских података*

Над овом табелом могуће је извршити **брисање примерка**. Брисање примерака се релизује селекцијом примерка који треба да се обрише у табели и притиском на типку <Delete> на тастатури. Ова функција дозвољена је само библиотекарима са посебним привилегијама.

Селекцијом једног примерка у табели примерака тај примерак се учитава у доњи део екранске форме и могућа је модификација података за селектовани примерак. На слици 5.21 приказана је екранска форма за унос локацијских података са једним учитаним примерком.



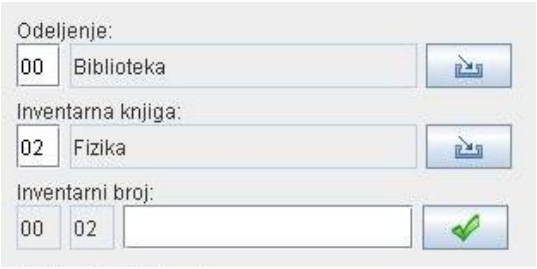
Слика 5.21 *Екранска форма за унос локацијских података са учитаним примерком*



Слика 5.22 *Шифарник статуса*

Подаци о једном примерку подељени су у три групе при чему за сваку групу постоји посебна картица, то су *Inventar*, *Signatura* и *Nabavka*. Приликом уноса података о примерку разликујемо две врсте поља за унос података, то су шифрирана и нешифрирана поља. Поље за унос шифриране вредности састоји се од поља за унос шифре, поља за испис значења те шифре и дугмета са иконицом за отварање шифарника. Шифарник се може отворити и пречицом са тастуре – тастер <F9>. Шифарници који се користе у овој екранској форми дефинишу се на нивоу библиотеке, за разлику од шифарника који се користе у библиографским записима који су дефинисани стандардом. На слици 5.22 приказан је шифарник статуса за библиотеку Департмана за математику и информатику.

Посебно поље за унос има инвентарни број (слика 5.23). Прве две цифре у инвентарном броју представљају шифру одељења и превлаче се из одговарајућег поља. Друге две цифре у инвентарном броју су шифра за инвентарну књигу која се превлачи из поља за унос инвентарне књиге. Остатак инвентарног броја је број од седам цифара и представља редни број књиге у неком низу. Дугме са иконицом омогућава генерисање новог инвентарног броја за изабрано одељење и инвентарну књигу.



Слика 5.23 *Унос инвентарног броја*

Притиском на дугме *Sačuvaj*, уколико се ради о уносу новог примерка он се складишти у релациону базу података и у табели примерака на екранској форми. Уколико је извршена модификација примерка све измене се складиште у базу података и у табели примерака на екранској форми. Дугме *Poništi* иницијализује екранску форму након чега се може вршити унос потпуно новог примерка. Дугме *Raspodela* отвара прозор за расподелу примерака по одељењима.

5.3.6 Пречице за брзи приступ командама

На панелу за брзи приступ командама налази се неколико пречица од којих свака извршава посебну акцију над записима. Свака од тих акција може се извршити и одређеном пречицом са тастатуре. У табели 5.1 дато је објашњење свих команди заједно са одговарајућим пречицама са тастатуре.

Табела 5.1 *Команде за рад са записима*

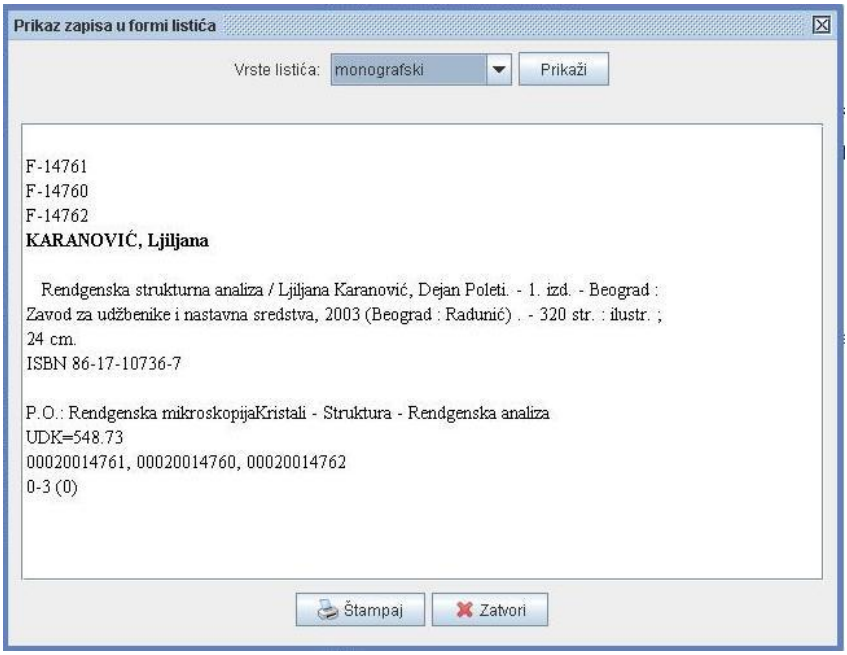
Иконица	Пречица са тастатуре	Значење
	<Ctrl>+<N>	Отварање новог записа у едитору
	<Ctrl>+<S>	Снимање записа
	<NumPad-7>	Додавање поља у стабло формата
	<Ctrl>+<V>	Провера валидности записа
	<Ctrl>+<L>	Отварање прозора за приказ листића
	<Ctrl>+<T>	Промена типа обраде

Приликом снимања записа, врши се провера његове валидности. Уколико запис није валидан (на пример нису унета сва обавезна потпоља) он не може бити снимљен и кориснику се пријављује одговарајућа порука.

Корисник-библиотекар може у току рада да провери валидност свог записа. Након позива одговарајуће акције кориснику се приказује извештај о валидности записа. На слици 5.24 приказане су поруке које се пријављују као извештај о валидности записа. На слици лево приказана је порука у случају да је запис валидан а десно је приказана порука која се пријављује у случају да запис није валидан.

Слика 5.24 *Извештаји о валидности записа*

На слици 5.25 приказан је прозор за приказ записа у форми листића. За сваку библиотеку дефинисан је падразумевани листић који се приказује приликом отварања овог прозора. Постоји могућност да се из падајуће листе са врстама листића изабере нека друга врста листића и притиском на дугме *Prikaži* на белој површини форме приказује за запис у изабраној форми листића.



Слика 5.25 *Приказ записа у форми листића*

Литература

[AACR] T. Desley. *The Logical Structure of the Anglo-American Cataloguing Rules - Part I*. The Joint Steering Committee for Revision of AACR, 1998 , http://www.collectionscanada.ca/jsc/aacrdel2.htm
(30. октобар 2006.)

[Apache] The Apache Software Foundation, http://www.apache.org
(12. april 2007.)

[BiblioML] *BiblioML Project*, Ministère de la culture et de la communication, France. http://xml.coverpages.org/biblioML.html
(19. април 2007.)

[BISIS03] Д. Сурла, З. Коњовић, Б. Пуповац, Б. Милосављевић, М. Видаковић, Т. Тошић, Т. Зубић. *Упутство за коришћење библиотечког софтверског система БИСИС вер. 3*, Група за Информационе технологије, Нови Сад, 2003.

[BISIS04] Монографија *Дистрибуирани библиотечки информациони систем БИСИС*, уредници Д. Сурла, З. Коњовић, Група за Информационе технологије, Нови Сад, 2004.

[BookMARC] BookMARC, Bibliographic Information Service http://www.bookmarc.pt/ (08. октобар 2007)

[BOOKSYS] Book Systems, Inc, http://www.booksys.com
(28. септембар 2006.)

[Budimir04] Будимир, Г., Сурла Д., *Систем за контролу квалитета XML библиографских записа*, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2004.

[CANMarc] *CAN/MARC*,
http://www.collectionscanada.ca/marc/index-e.html (30. октобар 2006.)

[CONCOURSE] Concourse Software Product,
http://www.booksys.com/v2/products/concourse/ (30. октобар 2006.)

[Dimić07a] Димић, Б., *XML едитор за обраду библиографске грађе*, магистарска теза, Природно-математички факултет, Нови Сад

[Dimić07b] Димић, Б., Милосављевић Б., *Контрола библиографских записа по UNIMARC стандарду*, CD-ROM YU INFO 2007, Копаоник 2007.

[Eclipse] Eclipse - an open development platform, http://www.eclipse.org (11. април 2007.)

[GISSystems] GIS Information Systems, http://www.gisinfosystems.com (12. октобар 2006.)

[ISBD] *Family of ISDBs: Publication list.* International Federation of Library Associations and Institutions, 2003. http://www.ifla.org/VI/3/nd1/isbdlist.htm. (30. октобар 2006.)

[ISIS] CDS/ISIS database software, http://www.unesco.org/isis/ Монографија *Дистрибуирани библиотечки информациони систем БИСИС*, уредници Д. Сурла, З. Коњовић, Група за Информационе технологије, Нови Сад, 2004.

[ISO2709] *Information and documentation - Format for information exchange*. International Organization for Standardization, 1996.

[Jakarta] The Apache Jakarta Project, http://jakarta.apache.org (13. април 2007.)

[Jakšić04] Jakšić, M., Mijić, V., Surla, D., *XML dokument UNIMARC formata*, PMF, Univerzitet u Novom Sadu, 2004.

[Java] Java Technology, http://java.sun.com (11. април 2007.)

[JavaIO] java.io (Java Platform SE 6) http://java.sun.com/javase/6/docs/api/java/io/package-summary.html (27. септембар 2007)

[JavaSQL] java.sql (Java Platform SE 6) http://java.sun.com/javase/6/docs/api/java/sql/package-summary.html (27. септембар 2007)

[JavaUTIL] java.util (Java Platform SE 6) http://java.sun.com/javase/6/docs/api/java/util/package-summary.html (27. септембар 2007)

[JTree04] How to Use Trees, http://java.sun.com/docs/books/tutorial/uiswing/components/tree.html (19. април 2007.)

[LOC] *The Library of Congress*,. http://www.loc.gov (02. октобар 2006.)

[Lucene] Lucene http://lucene.apache.org (13. април 2007.)

[MarcEdit] *MarcEdit*, http://oregonstate.edu/~reeset/marcedit/html/ (02. октобар 2006.)

[MARCFormat] *MARC Format Overview*. Network Development and MARC Standards Office, Library of Congress http://www.loc.gov/marc/status.html. (08. октобар 2006.)

[MARCRecsrvs] *MARC Record Services*, http://www.loc.gov/marc/marcrecsvrs.html (25. септембар 2006.)

[MARCService] *MARC Records, Systems and Tools*, http://www.loc.gov/marc/marcservice.html (25. септембар 2006.)

[MARCStandards] *MARC Standards*. Network Development and MARC Standards Office, Library of Congress, http://lcweb.loc.gov/marc/ (16. септембар 2003.)

[MARCSystems] *Marc Systems*, http://www.loc.gov/marc/marcsysvend.html (25. септембар 2006.)

[MARCTools] *MARC Specialized Tools*, http://www.loc.gov/marc/marctools.html (25. септембар 2006.)

[MARCXML] *MARC 21 XML Schema (MARCXML), Official Web Site,* Library of Congress, 2003, http://www.loc.gov/standards/marcxml (19. април 2007.)

[MARCXMLSchema] *MARC 21 XML Schema*, http://www.loc.gov/standards/marcxml/schema/MARC 21slim.xsd (17. април 2007.)

[Mijić03] В. Мијић. *XML едитор за опис UNIMARC формата,* магистарска теза, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2003. http://diglib.ns.ac.yu/ndltd/docs/set1/ndltd133/MijicVMagistarskaTeza.pdf

[Milosavljević07] Milosavljević B., Dimić B.*, XML schema of UNIMARC format variant and bibliographic record in BISIS software system*, NSJOM, Vol. 37, No.1, 2007, pp 115-128.

[MySQL] MySQL, http://www.mysql.com (12. април 2007.)

[OAI] Open Archives Initiative, http://www.openarchives.org/ (18. април 2007.)

[OAIXMLSchema] http://www.openarchives.org/OAI/oai_marc.xsd (18. април 2007.)

[Rađenović06a] Rađenović J., Milosavljević B., Surla D., *Generating catalog cards by the BISIS library software*, INFOTEKA, 7(2006)1-2, str. 61-74

[Rađenović06b] Рађеновић Ј., Сурла, Д., Милосављевић Б., *Софтверски пакет за генерисање библиотечких каталошких листића* (монографија), Иновациони центар за електронске библиотеке и архиве, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2006.

[SAX] SAX, http://www.saxproject.org/ (12. април 2007.)

[Škrbić04] Škrbič, S., Surla D., *Upravljanje XML bibliogarfskim zapisima u XML tehnologiji,* XXXI Simpozijum o operacionim istraživanjima SYM-OP-IS 2004, Iriški Venac 2004. pp133-136

[Škrbić05] Škrbić, S., Surla D., *Obrada bibliografske građe u XML native tehnologiji*, PMF, Univerzitet u Novom Sadu, 2005.

[TERAText] TeraText Database System, http://www.teratext.com (02. октобар 2006.)

[Tešendić] Тешендић, Д. *Софтверски систем за коришћење библиотечке грађе*, магистарска теза, Природно-математички факултет, Нови Сад, 2007.

[UKMARC] *The UKMARC Exchange Record Format*. The British Library, National Bibliographic Service. http://www.bl.uk/services/bibliographic/marc/marcexchange.html (19. април 2007.)

[UNIMARC94] *UNIMARC Manual: Bibliographic Format/ International Federation of Library Associations and Institutions.* IFLA Universal Bibliographic Control and International MARC Programme, New Providence, London 1994. http://www.ifla.org/VI/3/p1996-1/sec-uni.htm (19. април 2007.)

[USMARC] *USMARC INFORMATION*, http://www.info.library.yorku.ca/techserv/usmarc.htm (30. октобар 2006.)

[Vidaković04]. Видаковић, М., *Обрада библиографске грађе*, Дистрибуирани библиотечки информациони систем *БИСИС* (уредници Д. Сурла, З. Коњовић), Група за Информационе технологије, Нови Сад, 2004. стр. 65-81.

[Vulić] Вулић, Т., Билић, Г., *Ретроспективна конверзија каталошких листића у формат YUMARC,* Природно-математички факултет, Институт за математику, Нови Сад, 1997.

[XMLSchema] *XMLSchema*, http://www.w3.org/XML/Schema (30. октобар 2006.)

[XMLSpy] XMLSpy - XML editor for modeling, editing, transforming, & debugging XML technologies http://www.altova.com/products/xmlspy/xml_editor.html (20. април 2007)

[Z39.2] *Information interchange format*. National Information Standards Organisation, 1994 http://www.niso.org/standards/resources/**Z39-2**.pdf (19. апрул 2007.)

[Z39.50] *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*. National Information Standards Organisation, 2002. http://www.niso.org/standards/resources/Z39-50-200x.pdf
(19. април 2007)

[Zeremski01a] Zeremski, M., Surla, D., Modeliranje *UNIMARC formata u XML – Data jeziku,* Zbornik radova sa Simpozijuma o računarskim naukama i informacionim tehnologijama YU INFO 2001 (na CD-ROM-u), Kopaonik, 2001.

[Zeremski01b]Zeremski, M., Surla, D., *Modeliranje UNIMARC formata pomoću XML Schema jezika,* Zbornik radova sa XXVIII Jugoslovenskog simpozijuma o operacionim istraživanjima SYM-OP-IS 2001, Beograd, 2001. str. 261-264.

[Zeremski01c] Зеремски, М**.**, Сурла, Д., *Моделирање библиографских записа у облику XML докумената,* ИНФОТЕКА: Часопис за информатику и библиотекарство, Београд, 2(2001) 1-2, стр. 93-97.

[Zeremski01d] Zeremski, M., Surla, D., *Modeliranje bibliografskih zapisa pomoću XML Schema jezika,* Zbornik radova sa XXVIII Jugoslovenskog simpozijuma o operacionim istraživanjima SYM-OP-IS 2001, Beograd, 2001. str. 265-268.

[Zeremski02]Zeremski, M., Surla, D., *Validacija bibliografskih zapisa pomoću XML Schema jezika,* Zbornik radova sa Simpozijuma o računarskim naukama i informacionim tehnologijama YU INFO 2002 (na CD-ROM-u), Kopaonik, 2002.

[Zeremski04] Zeremski, M., Surla, D., *Validation of XML bibliographic records in Java environment,* Novi Sad Journal of Mathematics, 2004.

CIP – Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

025.3:007]:004.439XML

ДИМИЋ, Бојана
Obrada bibliografske građe u softverskom sistemu BISIS / Bojana Dimić, Dušan Surla. – Novi Sad : Prirodno-matematički fakultet, Departman za matematiku i informatiku, 2007 (Novi Sad : ABM ekonomik). - 101 str. : graf. prikazi ; 24 cm

Tiraž 150. – Bibliografija. – Abstract.

ISBN 978-86-7031-133-6
1. Сурла, Душан
а) Библиотечки информациони системи – Каталогизација – Програмски језик "XML"

COBISS.SR-ID 225880583

Мр Бојана Димић је истраживач приправник Природно-математичког факултета у Новом Саду. Рођена је (1982) у Сомбору. Дипломирала је (2005) на Природно-математичком факултету у Новом Саду, где је (2007) одбранила и магистарску тезу. Од марта 2007. године запослена је на Природно-математичком факултету у Новом Саду и ангажована је на пројекту Апстрактни модели и примена у рачунарским наукама (бр. 144017) који финансира Министарство науке и заштите животне средине Републике Србије. Има пет објављених научних радова који припадају области пројектовања библиотечких информационих система.

Др Душан Сурла је редовни професор Природно-математичког факултета у Новом Саду. Рођен је (1942) у Омсици, Република Хрватска. Дипломирао је (1969) на Филозофском факултету у Новом Саду. Магистрирао је (1976) на Машинском факултету у Новом Саду, а докторирао (1980) на новосадском Факултету техничких наука. На Природно-математичком факултету у Новом Саду ради од 1976. а за редовног професора изабран је (1991) за научну област информатика. Објавио је око 200 библиографских јединица из области роботике и области информациони системи.

ISBN 978-86-7031-133-6