UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Petra Cimerman

Kako in zakaj narediti svoj repozitorij

Diplomsko delo

Ljubljana, 2011

UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Petra Cimerman

Mentor: doc. dr. Janez Štebe
Somentorica: asist. dr. Nataša Kejžar

Kako in zakaj narediti svoj repozitorij

Diplomsko delo

Ljubljana, 2011

Zahvala

Z diplomskim delom se zaenkrat zaključuje obdobje mojega študijskega življenja. Zato bi se ob tej priložnosti najprej iskreno zahvalila prof. doc. dr. Janezu Štebetu in asist. dr. Nataši Kejžar, da sta me z veseljem sprejela pod svoje mentorstvo in me spretno vodila v pravo smer.

Zahvalila bi se rada tudi svoji družini, še posebej mami in očetu, ki sta mi omogočila študij in mi z vso ljubeznijo in potrpljenjem stala ob strani v vseh trenutkih.

Predvsem pa se zahvaljujem Luki Kralju za vso moralno in ostalo pomoč pri študiju in izdelavi diplomskega dela.

Kako in zakaj narediti svoj repozitorij

Število podatkov in informacij dandanes drastično narašča, za njihovo shranjevanje, vzdrževanje in organiziran dostop pa so potrebne posebne računalniške aplikacije, ki jih imenujemo digitalni repozitoriji. V diplomskem delu smo preučili, kako delujejo repozitoriji in njihove funkcije, in testirali, kako poteka izdelovanje lastnega repozitorija. Diplomsko delo je razdeljeno na tri vsebinske dele, in sicer na teoretični del, kjer smo na kratko opisali repozitorije, njihovo funkcionalnost in predstavili različne namene njihove uporabe. Sledi predstavitev enega od obstoječih primerov repozitorijev, to je sistema za nadzor izvedb *Subversion*. V zadnjem, praktičnem delu pa je predstavljen še lasten prototip repozitorija oz. vzorčna aplikacija, ki predavateljem in študentom pri predmetu *Informacijski viri in sekundarni podatki* omogoča shranjevanje gradiv (seminarske in domače naloge, literature, primerov izpitnih vprašanj ...), potrebnih za ta predmet, na enem mestu. Gre za repozitorij, kamor lahko predavatelji in študenti shranjujejo gradiva, ki jih lahko obnavljajo, pri čemer se gradiva ustrezno verzionirajo, njihove starejše različice pa se ohranijo in se do njih lahko kadar koli dostopa. V času nastajanja diplomskega dela smo spoznali, da razvoj lastnega repozitorija ni vedno najboljša rešitev. Prav tako obstaja že veliko raznovrstnih repozitorijev, tudi odprtokodnih ali brezplačnih, ki lahko zadovoljijo potrebe skoraj vsakogar.

Ključne besede: repozitorij, e-izobraževanje, sistem za nadzor izvedb, Subversion, vzorčna aplikacija.

How and why to build your own repository

Lately, the amount of data and information has been increasing drastically. To store, maintain and access them in an organized way, special application software is needed, particularly the repositories. In the present thesis, we undertook a research on the functioning of repositories and tried to build our own repositories. The thesis is divided in three topics. The first theoretical part comprises of a short presentation of repositories and their functionality as well as of various purposes they fulfil. In the second part, we present one of the existing repositories – the *Subversion*, revision control system. The last topic is a presentation of our own example of an application or a prototype of a repository that would enable data storage to lecturers and students at the course *Information sources and secondary data* to provide them with the information they need all in one place (storage of seminar papers, homework assignments, literature, examples of exam questions, etc.). This is a repository into which both lecturers and students can store the materials that can later be renewed and versioned, with older versions remaining stored and accessible at any time. During the process of the work on the thesis, we discovered that building your own repository is not always the best solution. There are numerous and diverse repositories already established, many of them open source or free applications that can meet the needs of almost every individual.

Keywords: repository, e-learning, revision control system, Subversion, example of an application.

Kazalo

1 Uvod

V informacijski družbi sta internet ter digitalni dostop do podatkov in informacij postala neprecenljivega pomena. Podatki in informacije se hranijo v digitalni obliki ali se ustvarjajo s pretvorbo iz analognih oblik z namenom čim lažjega in takojšnjega dostopa. Posledično število podatkov in informacij drastično narašča, za njihovo shranjevanje, vzdrževanje in organiziran dostop pa so potrebne posebne računalniške aplikacije, ki jih imenujemo digitalni repozitoriji. Organizacija oz. institucija, lahko tudi posameznik, se običajno odloči, da bo zgradil repozitorij, kjer bodo shranjeni dokumenti v digitalni obliki in bodo na voljo različnim uporabnikom (Kavčič - Čolić in Šmid 2007, 116).

V diplomskem delu predstavljamo repozitorije, predvsem to, kako lahko sami v praksi razvijemo oz. sprogramiramo repozitorij, kar pokažemo tudi na primeru. Ideja se je razvila med diskusijo s sošolci o tem, kako poenostaviti uporabo gradiv za posamezne predmete (seminarske naloge, vaje, domače naloge, literaturo, navodila za opravljanje obveznosti pri predmetu, primeri izpitnih vprašanj …). Študentom se namreč velikokrat zgodi, da imajo več različic teh gradiv na različnih računalnikih in prenosnih medijih (USB-ključi, prenosni diski, CD-ji …), dejansko pa ne vedo zagotovo, katera različica je zadnja, kar lahko ustvari veliko zmedo. Ena od rešitev takšnega problema je repozitorij, kamor bi lahko predavatelji in študenti shranjevali gradiva, hkrati pa bi jih lahko enostavno obnavljali, pri čemer bi se gradiva ustrezno verzionirala, njihove starejše različice pa bi se ohranile in bi se do njih lahko kadar koli dostopalo.

Cilj diplomskega dela je raziskati in predstaviti repozitorije, namene njihove uporabe in funkcionalnost ter na praktičnem primeru prikazati način in smotrnost izdelave repozitorija, ki bi rešil zgoraj predstavljeni problem. Želeli smo **preučiti**, kako delujejo repozitoriji in njihove funkcije in katere funkcije je smotrno vključiti v lastnem repozitoriju. Hkrati smo želeli **testirati**, kako poteka izdelovanje lastnega repozitorija, in sicer vse od načrtovanja repozitorija in njegovih funkcionalnosti do njegovega razvoja. V ta namen smo razvili nekakšen prototip repozitorija oz. vzorčno aplikacijo, ki predavateljem in študentom pri predmetu *Informacijski viri in sekundarni podatki* omogoča shranjevanje gradiv, potrebnih za ta predmet, na enem mestu. Na nek način gre za prepletanje klasičnega izobraževanja in e-izobraževanja.

Diplomsko delo je razdeljeno na tri vsebinske dele, in sicer na teoretični del, predstavitev obstoječega primera repozitorija in praktični del. V drugem poglavju smo nekaj besed namenili predstavitvi repozitorijev in njihovih funkcij. Bolj podrobno smo izpostavili tiste funkcije repozitorijev, ki smo jih uporabili tudi v praktičnem delu. Sledi predstavitev namenov uporabe repozitorijev na različnih področjih, med njimi sta tudi e-izobraževanje in sistem za nadzor izvedb. V tretjem poglavju smo predstavili obstoječi primer repozitorija, in sicer sistem za nadzor izvedb *Subversion*. Gre za predstavitev enega iz med mnogih repozitorijev, ki je najbolj razširjen med razvijalci programske opreme oz. programerji. Zanj smo se odločili, ker ima funkcionalnosti, ki delujejo na način, ki ga je bilo vredno preučiti kot pomoč za razvoj lastnega prototipa. V četrtem poglavju smo predstavili vzorčno aplikacijo oz. prototip repozitorija, ki smo ga sprogramirali, in delovanje njegovih funkcij. V zaključku tega poglavja sledi še kratka evalvacija aplikacije. V zadnjem, petem poglavju sledijo sklepne ugotovitve.

2 Repozitoriji

V tem poglavju bomo najprej predstavili digitalne repozitorije in opisali funkcionalnosti, ki naj bi jih imel vsak repozitorij. V nadaljevanju bomo predstavili še pristope, kako jih izdelati, in na podlagi primerov opisali različne namene uporabe repozitorijev.

2.1 Kaj je repozitorij

> *Od samega začetka računalniško podprtega procesiranja podatkov obstaja težnja procesiranja in hranjenja vedno večje količine podatkov in informacij na računalnikih. Zlasti v okoljih z obsežnimi informacijskimi sistemi, kot so na primer statistični, je naraščala potreba po iskanju poti in sredstev za načine obravnave te hitro naraščajoče količine (statističnih) podatkov. Tehnološki napredek in potrebe uporabnikov so končno pripeljale ne samo k izredno velikim podatkovnim bazam in njihovi razdelitvi v omrežja, ampak tudi do potrebe ustvariti in predstaviti natančno določena orodja za obravnavanje njihove vsebine, torej vsebine podatkov in informacij* ... (Vipavc in Klep 2003, 536).

Eno takšnih orodij je repozitorij. Ambrožič in drugi (2006, 3) dobro povzemajo bistvene opredelitve repozitorija oz. digitalne zbirke podatkov in informacij, ki je po njihovem mehanizem za upravljanje in shranjevanje digitalnih vsebin. Z drugimi besedami, gre za računalniško aplikacijo, ki je večinoma kompleksna relacijska baza, namenjena skladiščenju, dostopanju in upravljanju digitalnih dokumentov in podatkov (npr. institucionalni repozitorij univerz).

Repozitorij je lahko glede na njegovo osredotočenje tematski ali institucionalni. Vnašanje/vključevanje vsebine v institucionalni repozitorij osebju in institucijam omogoča kakovostno vodenje in ohranjanje vsebine, zato se lahko z njim podpira raziskovalne, učne in administrativne procese. Za zagotavljanje iskanja in dostopanja do vsebine se uporablja odprte standarde, ki hkrati omogočajo tudi poznejšo uporabo te vsebine (Repositories Support Project 2010c).

2.1.1 Kratek zgodovinski pregled

Programska podpora za izgradnjo in vzdrževanje repozitorija ter za zagotavljanje dostopa do njegove vsebine še ne zagotavlja tudi trajnega ohranjanja digitalnih vsebin. V nasprotju s tem so nekateri strokovnjaki digitalni repozitorij opredelili kot sistem, ki zagotavlja trajno ohranjanje in zaščito ter dostop do digitalnih objektov, za kar v strokovni literaturi velikokrat naletimo na izraz 'digitalni arhiv' (Ambrožič in drugi 2006, 3).

Kot navajata Kavčič - Čolić in Šmid (2007, 116), je bilo trajno ohranjanje digitalnih podatkov in informacij dolgo časa problem, zato je Mednarodna organizacija za standardizacijo (ISO) sprejela teoretični model za odprti arhivski informacijski sistem (OAIS, krajše za *Open Archive Information System*), ki zelo na splošno opisuje vse procese in elemente pri vnosu, arhiviranju in dostopu do digitalnih objektov, ki jih mora vključevati vsak digitalni arhivski sistem. Leta 2003 je bil teoretični model proglašen za ISO-standard (ISO-14721).

V primeru prototipa repozitorija, ki ga bomo razvili za namen preučitve izdelave lastnega repozitorija, ne bomo razmišljali o načinu trajnega ohranjanja podatkov in informacij, kajti za nas ni pomembno, da se podatke in informacije hrani daljše časovno obdobje (študenti bodo repozitorij uporabljali le toliko časa, da opravijo predmet). Kljub temu bomo v nadaljevanju uporabljali termin 'digitalni repozitorij' (oz. repozitorij) kot oznako za del digitalnega arhiva oz. aplikacijo, ki omogoča upravljanje, shranjevanje in dostop do shranjenega digitalnega objekta (Kavčič - Čolić in Šmid 2007, 117).

2.2 Funkcionalnosti repozitorijev

Kaj naredi digitalne repozitorije drugačne od drugih digitalnih zbirk? Heery in Anderson (2005, 1–2) menita, da se repozitoriji od drugih digitalnih zbirk razlikujejo v naslednjih točkah:

- Vsebina je objavljena v repozitorijih ne glede na to, ali jo objavi avtor vsebine, lastnik ali tretja oseba.

- Arhitektura repozitorijev upravlja tako vsebino kot tudi metapodatke[1]. Slednji so pomembni, ker izboljšujejo natančnost poizvedbe, zagotavljajo način (metodo) upravljanja elektronskih digitalnih objektov, lahko pomagajo zagotavljati verodostojnost podatkov in informacij ter so ključ interoperabilnosti.

- Repozitoriji nudijo nekaj osnovnih operacij, kot so uvoz (vnesti), izvoz (dobiti), iskanje in nadzor nad dostopom do vsebine (kdo lahko prebira, ureja ...).

- Repozitoriji morajo biti vzdržljivi in zanesljivi, dobro podprti in dobro vodeni.

Repozitoriji nudijo možnost, da se upravlja z vsebino in se jo lahko dolgoročno tudi ohranja. Shranjevanje vsebine v takšen sistem ji daje dodano vrednost, saj se jo lahko ponovno uporabi in obdela tudi večkrat, medtem ko vsebine v analogni obliki ni tako preprosto večkrat uporabiti ali obdelati. Ena od prednosti repozitorijev je ta, da je lahko vsak del vsebine opisan v nekaj podrobnostih s pomočjo metapodatkov. To deluje kot katalоški zapis v knjižničnem vodstvenem sistemu in omogoča iskanje po ključnih besedah znotraj repozitorijev (Repositories Support Project 2010c).

Na naslednjih straneh bomo na kratko predstavili posamezne zaželene funkcije vsakega repozitorija, kot so shranjevanje in zbiranje vsebine, zaklepanje dokumentov, uporaba OAI-protokola za zbiranje metapodatkov ...

Shranjevanje in upravljanje vsebine

Repozitoriji so lahko ena ali več povezanih podatkovnih baz različnih vrst, in sicer so lahko to relacijske baze, datotečni sistemi, XML-datoteke ... Nuditi morajo možnost shranjevanja besedil, vsebinskih blokov, binarnih podatkov in datotek ter metapodatkov. Hkrati pa morajo omogočati možnost **dostopa do vsebine**, ki jo shranjujejo, in nuditi možnost **poizvedb** po metapodatkih in vsebinskih blokih (Marjetica.net 2010).

Upravljanje vsebine vsebuje naloge, kot so varovanje vsebine in dodeljevanje pravic dostopa; urejanje uporabnikov; pregledovanje in urejanje statusa vsebine ter vpogled v delovne procese in beleženje sprememb ter omogočanje povrnitve v predhodno stanje (Marjetica.net 2010).

[1] Metapodatek (angl. *Metadata*) je opisni podatek ali kontekstualna informacija o že obstoječem podatku. Običajno so v obliki strukturnega niza elementov, ki opisujejo vir informacije oz. vsebino. Metapodatke lahko razvrstimo v več različnih skupin: opisni, tehnični, administrativni in varnostni metapodatki.

Za brezhibno delovanje celotnega procesa zbiranja, shranjevanja in objavljanja so potrebna orodja, procedure in človeški viri ... Komponente repozitorijev, ki skrbijo za delovni proces, ustvarijo in skrbijo za verigo dogodkov okrog zbiranja, shranjevanja in objavljanja (Marjetica.net 2010).

Bistvo vsakega repozitorija je zbiranje in shranjevanje vsebine, in sicer gre pri našem prototipu za datotečni sistem, kjer se bo shranjevala vsebina, in XML-datoteke, kjer se bodo beležili metapodatki. V primeru da bi omogočili iskanje po vsebini in metapodatkih v repozitoriju, bi bilo bolj smiselno uporabiti relacijske baze, ki so bolj prilagodljive glede tega, vendar tudi bolj zahtevne in kompleksne. Kako bodo lahko uporabniki upravljali z vsebino, bo pogojeno z njihovimi vlogami, ki bodo predstavljene in opisane v nadaljevanju (glej podpoglavje 4.1.1).

OAI-protokol

Med zahtevami, ki naj bi jih izpolnjevali delujoči javni repozitoriji, je tudi podpora OAI. To je OAI-protokol (krajše za OAI-PMH *Open Archives Initiative Protocol for Metadata Harvesting*), ki ga je razvil Open Archives Initiative. Uporablja se za zbiranje metapodatkovnih opisov zapisov v različnih arhivih. OAI deluje na tipu odjemalec – strežnik, ki išče zahtevane podatke in informacije po posodobljenih zapisih o arhivu. Podatke in informacije se lahko išče glede na datumski okvir in po ključnih besedah, ki jih določijo ponudniki (Wikipedia 2006).

V lastnem prototipu ne bomo uporabili omenjenega protokola, saj bomo poenostavili beleženje metapodatkov. Kako se bodo beležili metapodatki v našem primeru, bo predstavljeno v nadaljevanju v podpoglavju 4.1.3.

Ključavnica oz. zaklepanje

Ena od želenih, če že ne bistvenih, funkcij repozitorijev, je 'ključavnica' (angl. *Lock*), ki se uporablja, kadar lahko več uporabnikov hkrati dostopa do neke baze podatkov. Funkcija preprečuje, da bi se podatki poškodovali, kadar želi več uporabnikov hkrati zapisovati v bazo. Vsak posameznik lahko upravlja le s tistimi zapisi oz. predmeti v bazi, do katerih ima začasno ekskluzivno pravico, ki mu jo omogoča *ključavnica,* in traja, dokler tega dela ne opravi in odda *ključavnice* (Wikipedia 2005).

Zaklepanje je tudi pri našem prototipu repozitorija ena bistvenih funkcij, kajti treba je zagotoviti, da bodo dokumenti (npr. seminarske naloge) zaščiteni pred izgubo kakršnih koli sprememb, v primerih ko bo lahko določen dokument urejalo in spreminjalo več uporabnikov (npr. študentov).

Prosti dostop

Prednost repozitorijev je možnost prostega dostopa. Prosto dostopni digitalni repozitoriji so spletne strani, kjer lahko avtorji ali njihovi posredniki objavijo dokumente (npr. šolske publikacije), ki so vsem dostopni za prebiranje. Oznaka prosti dostop se nanaša na brezplačno uporabo vsebine z oznako 'za vsakogar'. V takšnih repozitorijih registracija ali vpis za prebiranje določene vsebine ne bi smela biti potrebna, vendar vsi repozitoriji niso prosto dostopni. Nekateri zaščitijo dostop do določenih digitalnih vsebin, tako da so te dostopne le določenim uporabnikom (npr. registriranim uporabnikom …). Prosto dostopna literatura je v digitalni obliki dostopna na spletu, brezplačna in brez omejitev glede avtorske zaščite in licenc (Repositories Support Project 2010a).

V našem prototipu repozitorija tega ne bomo omogočili, vendar je prvotno mišljeno, da bi končni produkt uporabnikom (tj. profesorjem in študentom) omogočal dostop do repozitorija preko spleta. Hkrati bi lahko tudi omogočili, da bi bil repozitorij prosto dostopen, vendar je to zaradi same vsebinske zasnove prototipa (kot tudi repozitorija kot končni produkt) nesmiselno. Vsebinska zasnova prototipa bo predstavljena v nadaljevanju (glej podpoglavje 4.1.1).

Ključne operacije

Na splošno so ključne operacije, ki naj bi jih vključevali repozitorični programi, uvoz (angl. *Import*), izvoz (angl. *Export*), identifikacija/iskanje (angl. *Identify*), shranjevanje (angl. *Store*), dostopanje do vsebine (angl. *Retrieve digital content*) in opis vsebine (angl. *Metadata*) (JISC infoNet, 2011). V našem primeru prototipa bomo vključili vse naštete operacije, le identifikacije/iskanja ne, saj je v tem obsegu nepotrebna.

2.3 Pristopi za razvoj repozitorijev

Pri razvoju lastnega repozitorija se lahko odločimo med tremi glavnimi pristopi (Repositories Support Project 2010b):

- program lahko **razvijemo/sprogramiramo** sami.
- **uporabimo/namestimo** lahko **standardne pakete**,
- **repozitorij** lahko **'zakupimo'/najamemo pri zunanjih ponudnikih te storitve**.

V tabeli 2.1 je primerjava med vsemi tremi pristopi, in sicer so predstavljene njihove prednosti in slabosti ter popularnost posameznega pristopa.

Tabela 2.1: Primerjava med pristopi za razvoj repozitorijev

	Prednosti	**Slabosti**	**Popularnost pristopa**
Lasten razvoj repozitorija	- Popolni nadzor aplikacije. - Naročena rešitev, prirejena potrebam in zahtevam naročnika.	- Za vzdrževanje je potrebno osebje. - Vzdrževanje je dolgotrajno. - Težje posodabljanje. - Brez podpore.	- Pogumna odločitev. - Ni popularen pristop.
Standardni repozitorični paketi	- Že pripravljeni programi, ki so večinoma bolj funkcionalni. - Redno jih posodabljajo. - Večina teh paketov je brezplačnih.	- Ni nadzora nad izboljšavami. - Redne izboljšave lahko zahtevajo stalno preurejanje nastavitev.	- Najbolj popularen pristop. - Primeri: Fedora Commons, Dspace, ePrints.
Gostovanje repozitorija pri zunanjih ponudnikih	- Že pripravljena rešitev (program), ki je večinoma bolj funkcionalna. - Redno jih posodabljajo, ni težav s posodabljanjem. - Minimalne zahteve po osebju, ki bi bilo potrebno za vzdrževanje. - Hitra vzpostavitev repozitorija.	- Skrb glede zaščite digitalne vsebine. - Dodaten strošek. - Ni nadzora nad izboljšavami.	- Popularnost tega pristopa narašča.

Vir: Repositories Support Project (2010b).

Standardni repozitorični paket je po navadi **repozitorična programska oprema** (angl. *Software repository*). To je inštalacijski paket, ki na računalniku namesti določeno programsko opremo, s katero lahko uporabnik nato dostopa do in ureja repozitorij, katerega baza podatkov in informacij je shranjena na strežniku (lokalnem, spletnem ...). Ti programski paketi vsebujejo številna orodja, ki olajšajo delo z repozitoriji. Nekatere

repozitorije je mogoče upravljati le s točno določenim programom, drugi pa so 'odprte narave' in se do njih lahko dostopa s poljubnim programom. V nekaterih primerih je treba za 'najem' repozitorijev na spletu s prenosom programa plačati 'vstopnico', drugi pa so brezplačni (Wikipedia 2007).

V četrtem poglavju bomo predstavili praktičen primer repozitorija, ki smo ga sami razvili oz. sprogramirali. Pri razvoju repozitorija bomo uporabili prvi pristop.

2.4 Kje se lahko uporablja repozitorije

Digitalni repozitoriji lahko vsebujejo širok obseg vsebine za različne namene in uporabnike. Vsebina, ki se shranjuje v repozitorije, ni odvisna od tehnoloških in programskih zmožnosti, temveč od odločitve posamezne institucije ali skrbnika sistema. Vsebine so lahko raziskovalni produkti, kot so časopisni članki ali raziskovalni podatki, e-teze, e-izobraževalni predmeti in pedagoška gradiva, administrativni podatki ... (Repositories Support Project 2010c).

Repozitorije se lahko uporablja na različnih področjih, in sicer pri programiranju, pri e-izobraževanju, pri e-poslovanju, pri digitalnih knjižnicah, v statistiki ... V tem podpoglavju bomo predstavili primere uporabe digitalnih repozitorijev za različne namene na treh od zgoraj naštetih področjih.

2.4.1 E-izobraževanje in repozitoriji

V praktičnem primeru (poglavje 4) bo predstavljen nekakšen prototip repozitorija, ki je po funkcionalnosti zelo blizu simulaciji izobraževanja na daljavo in e-izobraževanja. Tukaj bomo zato nekaj besed namenili digitalnim repozitorijem, ki se uporabljajo pri e-izobraževanju. Najprej pa bomo na kratko prestavili e-izobraževanje.

E-izobraževanje je izobraževalni proces, ki za svoje delovanje uporablja informacijsko-komunikacijsko tehnologijo (IKT) in njene večpredstavnostne elemente, kot so predstavitve in obogateno besedilo, animirane sekvence, video, skupna uporaba namizja, komunikacija s predavateljem in ostalimi udeleženci, izvajanje testov, samotestiranje … Proces se izvaja ločeno ali znotraj sistemov vodenja izobraževanja (ELefANTC 2010).

Značilnosti e-izobraževanja (ELefANTC 2010):

- med tečaji študija je študent fizično ločen od predavatelja/tutorja,
- uporablja tehnologije za dostop do študijskega materiala,
- uporablja tehnologije za komuniciranje s predavateljem/tutorjem in ostalimi udeleženci izobraževanja,
- ponuja le določeno stopnjo podpore.

»Ker je študent ločen od predavatelja, je metoda učenja pri e-izobraževanju tesno povezana z metodologijo izobraževanja na daljavo ...« (ELefANTC 2010).

Sistemi za upravljanje učenja

Po navajanju Clarka (v Božeglav 2009, 31) je sistem za upravljanje učenja (angl. *Learning Management Systems – LMS*) programska oprema, ki podpira izobraževanje in učenje v učnem okolju. Za okolje obstaja več sopomenk, na primer virtualno učno okolje – VLE (angl. *Virtual Learning Environment*) ali pa CMS (angl. *Course Management System*) oziroma LCMS (angl. *Learning Course Management System*). V Sloveniji je najbolj poznan sistem za upravljanje učenja Moodle[2].

V teh sistemih je cilj digitalnih repozitorijev dostava vsebin, kot so znanstvena in tehnična poročila, članki in drugo raziskovalno gradivo prek že obstoječe infrastrukture. V Evropi je bil tak repozitorij vzpostavljen v okviru projekta DRIVER (angl. *Digital Repository Infrastructure Vision for European Research*) (CORDIS v Božeglav 2009, 31). »Med digitalne repozitorije se uvršča tudi sisteme za objavo elektronskih izobraževalnih gradiv. Tak primer je repozitorij e-Scolarship, ki uporabnikom omogoča enostaven in cenovno učinkovit dostop do izobraževalnih gradiv« (CDL v Božeglav 2009, 31).

Omembe vredni pa so predvsem digitalni repozitoriji učnih gradiv, ki združujejo standardiziran katalog izobraževalnih gradiv z dokumentnim sistemom. Uporabnikom omogočajo enostavno klasifikacijo izobraževalnih gradiv in ponujajo spletni prostor za njihovo shranjevanje. Nekateri repozitoriji urednikom in uporabnikom omogočajo tudi dodatno označevanje gradiv ter njihovo ocenjevanje, napredno iskanje po gradivih ... (Arnes v Božeglav 2009, 57).

[2] Oktobra 2008 je imelo v Sloveniji učilnice Moodle postavljenih 168 javno objavljenih in 44 anonimnih organizacij. (Božeglav 2009, 31)

2.4.2 Sistem za upravljanje elektronskih dokumentov

Primer repozitorija so tudi sistemi za upravljanje elektronskih dokumentov ali krajše EDMS (angl. *Electronic Document Management System*), preko katerih se izvršuje informacijska podpora za upravljanje z dokumenti. Tak sistem mora nuditi aktivno podporo procesu upravljanja dokumentov, in sicer od nastanka, pregleda, odobritve, distribucije do arhiviranja dokumenta (Jakovljević v Šverko 2006, 5–6).

Jedro sistema za upravljanje elektronskih dokumentov je dokumentni sistem oz. repozitorij, ki uporabnikom olajša delo z dokumenti (integracija z orodji za izdelavo, urejanje in sestavljanje dokumentov, izpisi na zahtevo). Prav tako skrbi za varnost dokumentov (elektronski podpis, pravice dostopa, avtentikacija uporabnikov), spremljanje njihovih sprememb (nadzor nad različicami, kontrola izpisov) in obveščanje uporabnikov o novih ali spremenjenih dokumentih (Šverko 2006, 9).

2.4.3 Sistemi za nadzor izvedb

Pogosto se zgodi, da dokument ne nastane naenkrat, ampak nastaja postopoma s spremembami oz. dopolnitvami. Po navedbah Miheliča (2010) predstavlja vsaka sprememba dokumenta njegovo novo izvedbo (angl. *Revision*). Če ne vodimo izvedb dokumenta, se predhodna izvedba dokumenta pri shranjevanju (v isto datoteko) prepiše oz. izgubi. Včasih ne želimo izgube predhodne izvedbe, še več, želimo hraniti prav vse izvedbe dokumenta, od njegovega nastanka do njegove trenutne izvedbe.

Nadzor nad izvedbami dokumenta lahko vodimo ročno, npr. tako, da hranimo različne izvedbe v datotekah z različnimi imeni. Takšen pristop ima vrsto slabosti, npr. zahteva izredno disciplino uporabnika; poimenovanje datotek, ki hranijo različne izvedbe, je dokaj zahtevno, izraba prostora na pomnilnem mediju ni ekonomična, ker ne izkorišča podobnosti med posameznimi izvedbami (Mihelič 2010).

Zaradi zgoraj navedenih razlogov se nadzora izvedb navadno ne izvaja ročno, razen v najenostavnejših primerih. Naprednejši nadzor izvedb nam omogočajo sistemi za nadzor izvedb (angl. *Revision control system*) (Mihelič 2010). Kot navajajo Collins - Sussman in drugi (2002–2005), gre za skladišče, ki je podobno običajnemu strežniku datotek, s to razliko, da si zapomni vse spremembe, ki so bile kadar koli narejene na datotekah in mapah. Strežnik sistema za nadzor izvedb hrani vse izvedbe datotek v neki mapi, ta

shramba pa se imenuje repozitorij. To omogoča pridobivanje starejše izvedbe datoteke in pogled skozi zgodovino, kako so se podatki in informacije spreminjale in kdo je naredil spremembe. Gabrijelčič (2010) poleg tega še navaja, da sistem omogoča vodenje več vzporednih razvojnih vej, kar olajša hkratno delo več razvijalcev na istem projektu.

Sistemi za nadzor izvedb omogočajo shranjevanje in obnavljanje izvedb iz skladišča; vodenje dnevnika sprememb; primerjanje med izvedbami; avtomatsko označevanje izvedb z različicami, z dokumenti pa lahko upravlja več oseb hkrati, pri tem pa ni možno, da bi kdo komu kdaj kaj prepisal oz. izgubil (Collins - Sussman in drugi 2002–2005). V naslednjem poglavju bomo opisali enega od obstoječih primerov sistema za nadzor izvedb *Subversion.*

3 Odprtokodni sistem za nadzor izvedb Subversion

V tem poglavju bomo na kratko predstavili sistem za nadzor izvedb *Subversion*, ki smo ga bolj podrobno preučili ter s tem pridobili nekaj idej in zamisli za razvoj lastnega repozitorija. Zakaj ravno *Subversion*? Njegove funkcije so dobro zastavljene, tako zaklepanje kot način verzioniranja se učinkovito izvajata. To je ključnega pomena pri razvoju našega prototipa.

Subversion (SVN) je eden najbolj priljubljenih odprtokodnih sistemov za nadzor izvedb. Je splošen sistem, ki omogoča urejanje *katere koli* zbirke datotek, vključno z datotekami izvorne kode programske opreme. Njegova glavna naloga je vodenje sprememb podatkov in informacij (Collins - Sussman in drugi 2002–2005). V nadaljevanju bomo opisali nekatere pomembne značilnosti sistema *Subversion*, na koncu poglavja pa bomo izpostavili še nekaj idej in zamisli za razvoj lastnega prototipa repozitorija.

3.1 Značilnosti sistema Subversion

- **Vodenje izvedb map -** *Subversion* ima 'virtualni' datotečni sistem pod nadzorom izvedb, ki upravlja spremembe na celotnem drevesu map. Pod nadzorom so datoteke *in* mape (Collins - Sussman in drugi 2002–2005).

- **Operacije**, ki jih *Subversion* omogoča nad datotekami (Küng in drugi 2010):
    - Uvozi (angl. *Import*) – uvoz datoteke v repozitorij.
    - Izvozi (angl. *Export*) – izvoz iz repozitorija na lokalni računalnik.
    - Prevzemi (angl. *Check-out*) – kot prejšnja funkcija, vendar tako, da je mogoče spremenjene datoteke ponovno 'oddati' v repozitorij.
    - Posodobi (angl. *Update*) – prevzemanje zadnje izvedbe datoteke iz repozitorija.
    - Spoji (angl. *Merge*) – združevanje sprememb iste datoteke v novo, združeno izvedbo.
    - Objavi (angl. *Commit*, *Submit, Check-in*) – shranjevanje spremenjene datoteke v repozitorij, tudi brisanje.
    - Pokaži dnevnik (angl. *History*) – zgodovina sprememb datoteke.

- Razlikuj (angl. *Diff*) – razlika med lokalno datoteko in ekvivalentno različico v repozitoriju ali med dvema poljubnima različicama iste datoteke.
- Okrivi (angl. *Annotate*) – izpis avtorjev posameznih vrstic neke različice.

- **Zaklepanje in združevanje** – vse lokalne datoteke so samo za branje, dovoljenje za pisanje pa ima hkrati lahko le en uporabnik. Vsak lahko datoteko lokalno spreminja, preden jo shrani v repozitorij, mora vanjo dodati spremembe, ki so jih medtem naredili drugi (Collins - Sussman in drugi 2002–2005).

- **Metapodatki pod nadzorom izvedb** – vsaka datoteka in mapa ima prirejeno nevidno množico 'lastnosti' (metapodatke). Lastnosti so pod nadzorom izvedb, prav tako kot vsebina datoteke (Collins - Sussman in drugi 2002–2005).

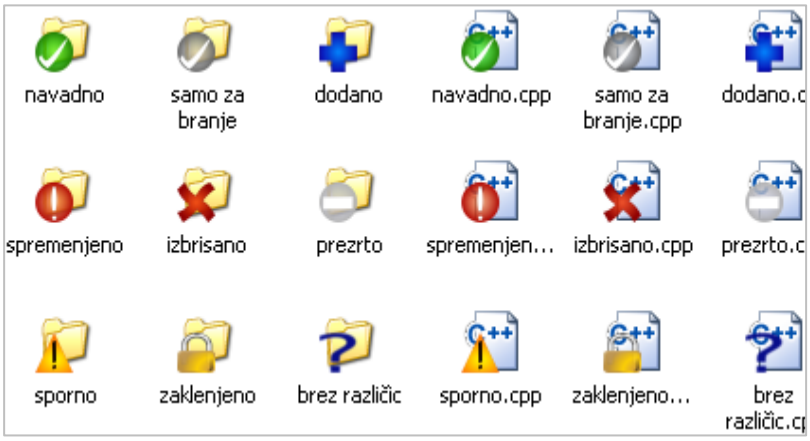
- **Različne možnosti dostopa do skladišča** – *Subversion* je vpeljal abstrakten koncept dostopa do skladišča, kar uporabnikom omogoča, da lahko dostopajo do skladišča tudi preko omrežja (internet, LAN, WAN) (Collins - Sussman in drugi 2002–2005).

- **Konsistentno upravljanje s podatki** – *Subversion* shrani originalno oz. izvorno datoteko, vse ostale spremembe (izvedbe) pa beleži tako, da zapisuje razlike med datotekami z dvojiškim algoritmom za razlikovanje, ki deluje tako na tekstovnih (uporabniku berljivih) kot na dvojiških (uporabniku neberljivih) datotekah. Datoteke obeh tipov so enako stisnjene in shranjene v skladišču, razlike pa se prenašajo v obeh smereh po mreži (Collins - Sussman in drugi 2002–2005).

- **Tortoise SVN** je brezplačen odprtokodni odjemalec oz. grafični uporabniški vmesnik za sistem nadzora izvedb *Subversion*. Slednji upravlja datoteke in mape skozi čas, ki pa so shranjene v centralnem *skladišču* (to je na strežniku *Subversion*) (Küng in drugi 2010, predgovor).

  Integrira se v lupino operacijskega sistema Windows in s tem omogoči, da se lahko upravlja datoteke in mape kar v Raziskovalcu ali katerem drugem upravljalniku datotek. Stanje vsake datoteke pod nadzorom izvedb nakazuje majhna prekrivna ikona (glej sliko 3.1).

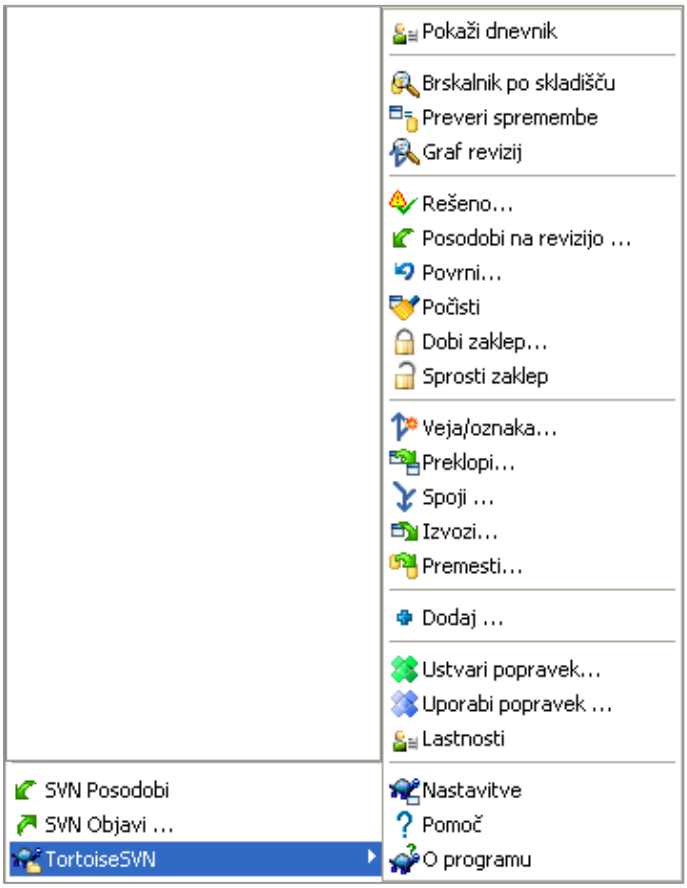
Slika 3.1: Prekrivne ikone map in datotek pod nadzorom izvedb



Vir: Küng in drugi (2010).

Vsi ukazi sistema *Subversion* so tudi na razpolago v kontekstnem meniju Raziskovalca (glej sliko 3.2).

Slika 3.2: Kontekstni meni za mapo pod nadzorom izvedb



Vir: Küng in drugi (2010).

3.1.1 Snovanje idej in konceptov za nastanek lastne vzorčne aplikacije

Po preučitvi sistema za nadzor izvedb *Subversion* in njegovih funkcij lahko izpostavimo nekaj idej in zamisli, ki nam bodo v pomoč pri nastajanju lastnega repozitorija. V spodnji tabeli je primerjava med funkcijami sistema *Subversion* in funkcijami našega prototipa.

Tabela 3.1: Primerjava funkcij sistemu *Subversion* in funkcij lastnega prototipa

Funkcije	Sistem za nadzor izvedb Subversion	Lasten prototip repozitorija
Vodenje izvedb map	Da	Da
Zaklepanje in združevanje	Da	Da, bomo nadgradili
Metapodatki pod nadzorom izvedb	Da	Ne
Različne možnosti dostopa do skladišča	Da, tudi preko interneta ...	Ne
Konsistentno upravljanje s podatki	Da	Da, vendar bolj preprosto
Integracija z lupino operacijskega sistema (kontekstualni meniji ...)	Da, Tortoise SVN	Ne
Operacije		
Uvoz	Da	Da
Izvoz	Da	Da
Prevzemi	Da	Da, združeni v eno
Posodobi	Da	
Spoji	Da	Ne
Objavi	Da	Da
Pokaži dnevnik	Da	Da
Razlikuj	Da	Ne
Okrivi	Da	Ne

Grafični vmesnik bomo razvili sami, pri čemer našega prototipa ne bomo integrirali z lupino operacijskega sistema Windows, kot to omogoča *Subversion* s **Tortoise SVN**, saj je to za nas enostavnejša rešitev. Z razvojem prototipa namreč želimo spoznati delovanje repozitorijev in njihovih funkcij, pri tem pa nam grafični vmesnik ni bistvenega pomena za doseganje želenega cilja. Tortoise SVN je nedvomno prednost sistema *Subversion* in ga je smiselno bolj preučiti ter razmisliti o podobnih rešitvah pri odločitvi o lastnem razvoju repozitorija.

Skladiščenje dokumentov se bo izvajalo lokalno na uporabnikovem računalniku, prav tako tudi vse ostale operacije za dostopanje in verzioniranje dokumentov. V primeru nadgradnje prototipa v končni produkt pa je smiselno **omogočiti dostop do skladišča** preko omrežja.

V prototipu bomo implementirali večino **operacij**, ki jih omogoča *Subversion*, razen *Spoji*, *Razlikuj* in *Okrivi*, ki jih ne potrebujemo v našem repozitoriju. Operaciji *Prevzemi* in *Posodobi* pa bomo združili v eno.

Tudi pri verzioniranju dokumentov bomo uporabili enostavnejšo rešitev, in sicer se bo dokument ob vsakem posodabljanju verzioniral v celoti in ne le njegove spremembe, kot to izvaja *Subversion*, ki **konsistentno upravlja s podatki**.

Zaklepanje dokumentov bomo izvedli podobno kot v sistemu *Subversion*, vendar bomo to lastnost še nadgradili, tako da bomo upoštevali tudi uporabnikove vloge in njihove pravice. Glede na vsebinsko zasnovo našega prototipa (glej podpoglavje 4.1.1) je to eden od razlogov, zakaj smo se lotili lastnega razvoja repozitorija, ki ga bomo predstavili v naslednjem poglavju.

4 Implementacija praktičnega primera repozitorija

V drugem in tretjem poglavju smo spoznali osnovne ideje in koncepte delovanja repozitorijev. V tem poglavju pa bomo poskušali večino teh funkcionalnosti uporabiti in implementirati v programsko rešitev. Najprej bomo nekaj besed namenili predstavitvi vsebinske zasnove, sledi pa še predstavitev tehnične zasnove prototipa oz. vzorčne aplikacije. V slednji bomo podrobno opisali delovanje posameznih funkcionalnosti. Na koncu tega poglavja bomo ocenili še uporabnost in smotrnost vzorčne aplikacije. Želeli smo razviti vzorčno, vendar preprosto aplikacijo, ki čim bolj zajema funkcionalnosti repozitorija, ki so omenjene v prejšnjih poglavjih.

4.1 Vsebinska in tehnična zasnova

Kot smo zapisali že v samem uvodu, je namen našega repozitorija, da se študentom olajša uporabo digitalnih gradiv, ki jih potrebujejo in uporabljajo v času študija, v našem primeru za uporabo gradiv pri predmetu *Informacijski viri in sekundarni podatki*. Glavna naloga repozitorija je, da lahko s pomočjo te aplikacije tako študenti kot tudi profesorji objavljajo, shranjujejo in prebirajo digitalne dokumente, ki se uporabljajo in ustvarjajo v času izvajanja omenjenega predmeta. Hkrati pa se ohranjajo vse izvedbe posameznega dokumenta, ki je bil dodan v aplikacijo. Tako se prepreči kakršna koli izguba podatkov in dokumentov, saj tega ni treba opravljati ročno.

Vzorčna aplikacija omogoča naslednje funkcije:

- prijavo uporabnikov;
- dodajanje in urejanje uporabnikov (možnost dodajanja večjega števila uporabnikov v enem koraku – z uvozom XML-datoteke);
- dodajanje in brisanje map;
- dodajanje, branje, urejanje in brisanje dokumentov;
- zaklepanje dokumentov – dokumenta istočasno ne more urejati več uporabnikov;
- verzioniranje dokumentov – beleženje vseh izvedb posameznega dokumenta;

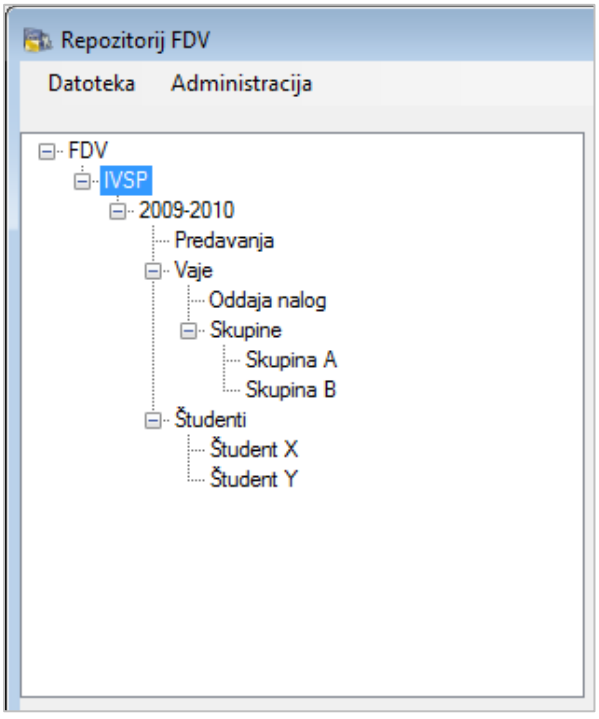
- zgodovina sprememb dokumentov – vpogled v vse izvedbe posameznega dokumenta;
- beleženje metapodatkov za posamezne dokumente in mape;
- dodeljevanje vlog uporabnikom (Administrator, Profesor in Študent). Vsaka vloga ima vnaprej definirane pravice nad posameznimi mapami. Pravice nad dokumenti pa so definirane z vlogo uporabnika in posamezno mapo.

4.1.1 Vsebinska zasnova

Informacijski viri in sekundarni podatki (IVSP) je obvezen predmet za študente *Družboslovne informatike* – visokošolski študijski program. Predmet se izvaja v drugem letniku, in sicer en semester, deli pa se na predavanja in vaje. Izvajata ga predavatelj in asistent. Obveznosti študentov pri tem predmetu so obvezna prisotnost na predavanjih in vajah, oddaja do 8 krajših izdelkov (domače naloge) ali izdelava krajše (lahko tudi skupinske) seminarske naloge in pozitivno opravljen pisni izpit (Fakulteta za družbene vede 2010).

Na podlagi zasnove predmeta IVSP želimo v vzorčni aplikaciji za vsako leto izvajanja tega predmeta imeti naslednjo drevesno strukturo map (glej sliko 4.1), ki ni v celoti vnaprej ustvarjena (več o tem v nadaljevanju):

Slika 4.1: Drevesna struktura map za predmet *Informacijski viri in sekundarni podatki*



V mapi *Predavanja* lahko profesor in asistent objavljata vse dokumente, ki so potrebni za izvajanje predavanj (npr. prosojnice, načrt izvedbe predmeta, pomožno literaturo v elektronski obliki, vprašanja za izpit ...). V mapo *Vaje* lahko profesor in asistent objavljata vse dokumente, ki so potrebni za izvajanje vaj (npr. prosojnice, navodila za opravljanje nalog na vajah, navodila za domače naloge …). Mapa *Oddaja nalog* je namenjena študentom za oddajanje domačih nalog ter profesorju in asistentu za objavljanje komentarjev in ocen oddanih domačih nalog.

Mapa *Skupine* je delovno okolje, kjer lahko študenti v skupinah izdelajo seminarske naloge. Vanjo naj bi profesor in asistent objavljala dokumente v zvezi s seminarskimi nalogami in skupinami (npr. navodila za izdelavo seminarske naloge, teme seminarskih nalog, obveznosti posamezne skupine ...). Profesor in asistent lahko v tej mapi ustvarita nove podmape za posamezno skupino in določita, kateri skupini pripada posamezen študent. Tako lahko imajo skupine na enem mestu zbrane vse dokumente, ki nastanejo med izdelavo seminarske naloge (npr. osnutek seminarske naloge, končno različico seminarske naloge, komentarje profesorja in asistenta, priloge, literaturo, zapiske iz konzultacij ...). Glavna prednost map posameznih skupin je, da lahko vsi njeni člani dodajajo in urejajo vse dokumente znotraj te mape, vendar jih lahko istočasno ureja samo en uporabnik, aplikacija pa poskrbi, da se hranijo vse njegove izvedbe, in s tem prepreči kakršno koli izgubo podatkov.

Študentom je na voljo še njihova lastna mapa, do katere lahko dostopajo v mapi *Študenti*. Mapa je poimenovano po študentovem imenu in priimku, študent pa lahko vanjo dodaja lastne dokumente (npr. zapiske, opravljene vaje …) in ustvarja nove mape.

Uporabniki lahko v repozitorij shranjujejo dokumente vseh vrst in formatov, kot so besedila (doc, pdf, TeX, txt, html ...), podatki (xls, sps…), slike (gif, png, jpg ...), predstavitve (ppt), videoposnetki (avi, mpeg, divx ...), programi (exe), izvorna koda (php, py, css ...), arhivski dokumenti (zip, rar ...) in ostalo.

Pravice uporabnikov glede na vlogo

Uporabniki imajo v aplikaciji različne vloge, ki določajo, kakšne pravice imajo v posameznih mapah in kakšne pravice imajo nad posameznimi dokumenti.

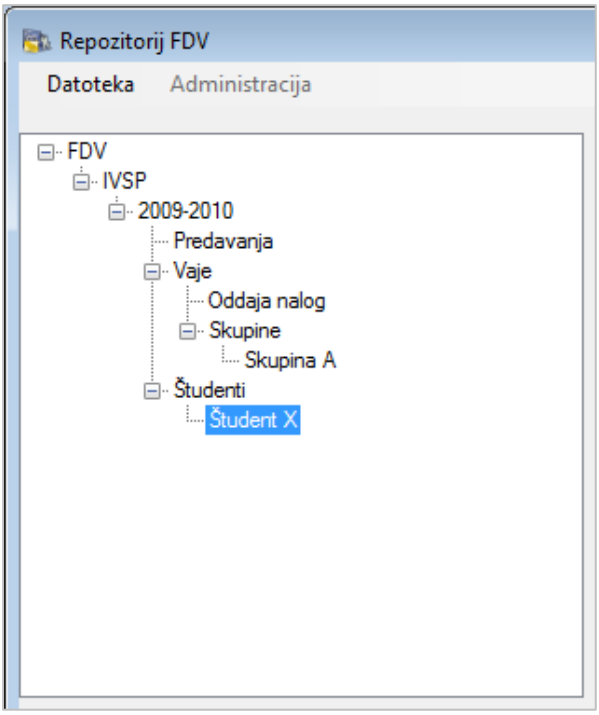
Uporabniki se razlikujejo glede na tri vrste vlog, ki so *Administrator*, *Profesor* in *Študent*.

- **Administrator** ima vpogled v vse mape, ki obstajajo v aplikaciji. Znotraj teh map vidi vse dokumente, ne more pa dodajati novih. Posameznega dokumenta ne more brati, urejati in brisati. Lahko briše samo mape in šele takrat tudi datoteke, ki so v njih shranjene. V zgornji orodni vrstici aplikacije je zavihek Administracija – Uporabniki, kjer lahko samo administrator ureja in dodaja uporabnike. Dodaja lahko ročno vsakega posebej ali pa jih uvozi več na enkrat iz vnaprej pripravljene XML-datoteke.

- **Profesor** – tako profesor kot tudi asistent imata vlogo *Profesor*. Uporabniki s to vlogo vidijo mape *Predavanja, Vaje, Oddaja nalog* in *Skupine*. V teh mapah lahko vidijo in berejo vse dokumente ter dodajajo nove. Urejajo in brišejo pa lahko samo tiste dokumente, ki so jih dodali uporabniki z isto vlogo. Dokumentov, ki so jih dodali uporabniki z vlogo *Študent*, torej ne morejo urejati in brisati. Uporabnik z vlogo *Profesor* (v nadaljevanju *Profesor*) lahko samo v mapi *Skupine* dodaja (in potem tudi vidi) nove mape za vsako skupino posebej. Pri dodajanju nove mape *Profesor* določi naziv (načeloma je to ime skupine) in uporabnike, ki vidijo to mapo (določi člane skupine). Člane izbere s seznama uporabnikov, ki imajo vlogo *Študent* in še niso člani nobene skupine. Znotraj mape posamezne skupine ima enake pravice kot v prej omenjenih mapah. Dokumente, ki so jih dodali *Profesorji*, lahko vidijo in berejo vsi uporabniki aplikacije.

- **Študent** vidi mape *Predavanja, Vaje, Oddaja nalog, Skupine* in *Študenti* ter znotraj slednjih dveh podmapo skupine, katere je član, in podmapo z imenom in priimkom študenta, ki je prijavljen. *Študent X*, ki je član *Skupine A,* bo tako videl drevesno strukturo map, kot jo prikazuje slika 4.2:

Slika 4.2: Drevesna struktura map, ko je prijavljen *Študent X*



V mapah *Predavanja, Vaje* in *Skupine* lahko uporabnik z vlogo *Študent* (v nadaljevanju *Študent*) vidi vse dokumente, vendar jih lahko samo bere. V teh mapah so le dokumenti, ki so jih dodali *Profesorji*. V mapi *Oddaja nalog* lahko prijavljen *Študent* dodaja dokumente, katerih ne more urejati in brisati, ko jih enkrat doda. Vidi in bere lahko samo lastne dokumente in dokumente, ki so jih dodali *Profesorji*. Ne more pa videti dokumentov, ki so jih dodali drugi *Študenti*. *Študent X* tako ne more prepisati domače naloge od drugih študentov, saj gre za individualno delo. V mapi skupine, katere član je *Študent*, lahko vidi in bere vse dokumente ter dodaja nove. Dokumente, ki jih je dodal sam ali drugi *Študenti*, lahko tudi ureja in briše. *Študent* v vseh teh mapah ne more ustvariti nove mape, prav tako jih ne more brisati. Za vsakega *Študenta* se ob njegovi prvi prijavi ustvari mapa z njegovim imenom in priimkom. To mapo vidi le ta *Študent*, v njem pa ima vse pravice (dodajanje, branje, urejanje in brisanje dokumentov ter dodajanje in brisanje map).

4.1.2 Uporabljena tehnologija

Za izdelavo aplikacije smo uporabili *.NET Framework 3.5 SP1*, ki je napisana v jeziku *C#*. Grafični vmesnik je narejen v tehnologiji *Win Forms*. Za razvojno orodje pa smo uporabili *MS Visual Studio 2008*.

C# je objektno orientiran programski jezik, ki ga je razvil *Microsoft*. Oblikovan je bil za delo z njihovim .NET ogrodjem (*.NET Framework*), ki je platforma oz. knjižnica, ki predstavlja ogrodje za vsa .NET orientirana programska orodja. Za pisanje programov v jeziku *C#* (in tudi ostalih jezikih v okolju .NET) je priporočljivo uporabiti razvojno okolje *MS Visual Studio*. Slednji združuje zmogljiv urejevalnik kode, prevajalnik, razhroščevalnik, orodja za dokumentacijo programov in druga orodja, ki pomagajo pri pisanju programskih aplikacij. (Lokar in Uranič 2008, 8)

Glede na to, za kakšno vrsto programske rešitve gre, je v okolju *MS Visual Studio* vnaprej pripravljenih več različnih tipov projektov, eden takšnih projektov je *Windows Forms Application*, ki je namenjen za gradnjo namiznih aplikacij s podporo grafičnih gradnikov (Lokar in Uranič 2008, 8).

Osnove programiranja smo pridobili med samimi študijem. Uporabe programskega jezika *C#* pa smo se naučili naknadno s pomočjo poznavalca. Pri razvoju aplikacije smo večino kode napisali sami, za grafični vmesnik pa je bil ta del kode generiran s pomočjo projekta *Windows Forms*. Aplikacija ima skupno 3421 vrstic kode, dejansko pa smo napisali 2012 vrstic kode.

4.1.3 Tehnična zasnova

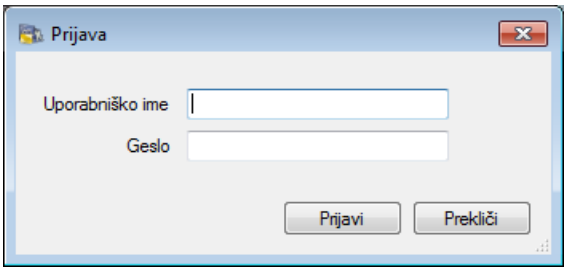
Diplomsko delo ima na zgoščenki priloženo še vzorčno aplikacijo *RepozitorijFDV* (glej prilogo Č). Za delovanje aplikacije je treba imeti na računalniku naložen operacijski sistem *Windows* in že omenjeni *.NET Framework 3.5 SP1*[3]. Aplikacijo se lahko zažene z dvojnim klikom na datoteko *RepozitorijFDV.exe*, ki se nahaja na priloženi zgoščenki v mapi *Priloga Č*.

Aplikacija je primarno namenjena zgolj prikazu delovanja repozitorija. Gre za prototip, zato se ob prvem zagonu ustvari mapa na 'C:\FDV\Repo\'. V našem primeru je ta mapa simulacija centralnega strežnika (v nadaljevanju *repozitorij*), kamor se ustvarjajo nove mape in se shranjujejo dokumenti ter metapodatki.

Za vstop v aplikacijo je potrebna predhodna prijava (angl. *Login*), saj se glede na vlogo posameznega uporabnika naloži glavno okno aplikacije (glej sliko 4.3).

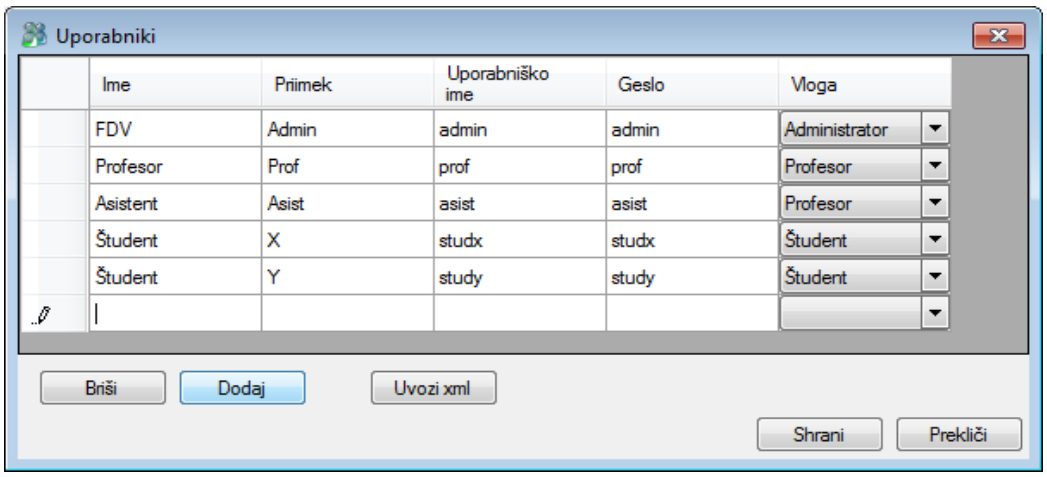
[3] Program je brezplačen, dostopen je na spletni strani http://www.microsoft.com/downloads/en/details.aspx?FamilyID=ab99342f-5d1a-413d-8319-81da479ab0d7. V primeru operacijskega sistema *Windows 7* namestitev tega programa ni potrebna.

Slika 4.3: Okno za prijavo



Prijava se uspešno izvede le, če je uporabnik že registriran, torej shranjen v XML-datoteki uporabnikov. V aplikacijo je bilo predhodno vnesenih pet uporabnikov z različnimi vlogami. Slika 4.4 prikazuje, kateri uporabniki so že dodani ob prvem zagonu aplikacije.

Slika 4.4: Okno za dodajanje in urejanje uporabnikov



Uporabnike se lahko dodaja ročno (vsakega posebej) ali z uvozom vnaprej pripravljene XML-datoteke, ki lahko vsebuje več uporabnikov (glej sliko 4.4). V prilogi A je primer vsebine takšnega XML-dokumenta.

Po uspešni prijavi se odpre glavno okno aplikacije (glej sliko 4.5), ki ima:

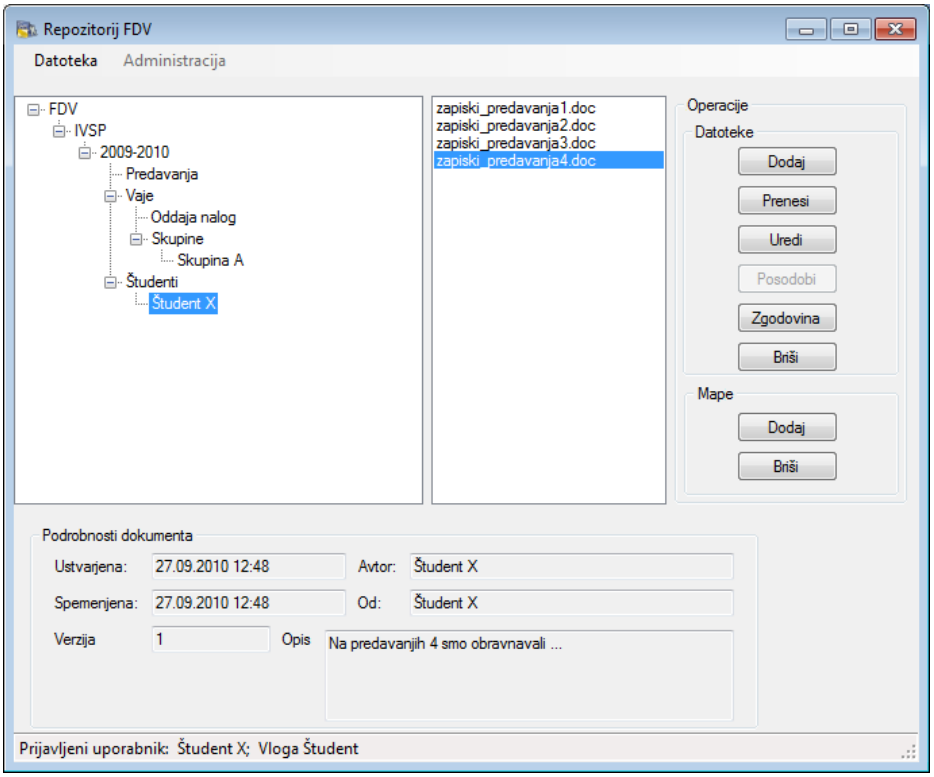
- na levi drevesno strukturo map;
- na sredini seznam dokumentov, ki se nahajajo v trenutno izbrani mapi;
- na desni skupini gumbov za izvajanje različnih operacij nad dokumenti in mapami,
- spodaj prostor za izpis lastnosti izbranega dokumenta.

Ob prvem zagonu aplikacije se v repozitoriju ustvarijo osnovne mape *Predavanja, Vaje* in *Študenti* ter znotraj mape *Vaje* še *Oddaja nalog* in *Skupine*. Vsaki mapi in tudi

dokumentu v aplikaciji pripada istoimenska skrita XML-datoteka, kamor se beležijo metapodatki. Za vse omenjene mape se med metapodatke beležijo naziv in vrsta mape ter podatki o pravicah. To je seznam vlog za branje in pisanje ter seznam uporabnikov, ki imajo pravice za branje in pisanje.

Ko se uporabnik uspešno prijavi in se mu odpre glavno okno aplikacije, se lahko loti dodajanja, branja in shranjevanja dokumentov. Desni del slike 4.5 prikazuje, katere operacije se lahko izvajajo nad dokumenti in mapami v aplikaciji.

Slika 4.5: Glavno okno aplikacije



Operacije, ki se izvajajo nad dokumenti

Z gumbom *Dodaj* se lahko dodaja nove dokumente. Ta se aktivira v izbrani mapi, ko ima prijavljen uporabnik pravico pisanja (dodajanje, urejanje in brisanje), ki je določena z njegovo vlogo. Uporabniku se odpre novo okno, kjer mora izbrati dokument na svojem računalniku, ki ga želi dodati, ter ga po lastni želji opisati. Ko uporabnik izvede vse zahtevane korake, aplikacija datoteko prenese (angl. *Copy*) v želeno mapo v repozitoriju, hkrati pa se med metapodatke zabeleži naziv, avtor, izvedba (različica: 1), čas in datum vnosa, čas in datum spremembe, avtor spremembe, ali je dokument

zaklenjen in kdo ga je zaklenil ter opis, ki ga je uporabnik napisal pri dodajanju dokumenta.

Gumb *Prenesi* se aktivira pri izbranem dokumentu, nad katerim ima prijavljen uporabnik pravico branja. Ob kliku na ta gumb se odpre novo okno, kjer uporabnik izbere mesto na računalniku, kamor želi shraniti izbrani dokument.

Pri gumbu *Uredi* se izvede popolnoma enak postopek kot pri gumbu *Prenesi,* le da mora uporabnik imeti v tem primeru pravico pisanja (urejanje). Postopek pa se ne bo izvedel, če izbrano datoteko že ureja drug uporabnik. S tem postopkom se med metapodatke izbrane datoteke namreč zabeleži, da **je dokument zaklenjen** s strani uporabnika, ki si je to pravico prvi pridobil (glej prilogo C). S tem se prepreči, da bi dokument istočasno urejalo več uporabnikov, saj lahko v nasprotnem primeru nastane popolna zmeda (npr. izguba podatkov ...). Dokument ima oznako *Zaklenjen,* dokler ga uporabnik ne posodobi (gumb *Posodobi*). To velja za vse dokumente v celotnem repozitoriju, ne glede na vlogo ali mapo.

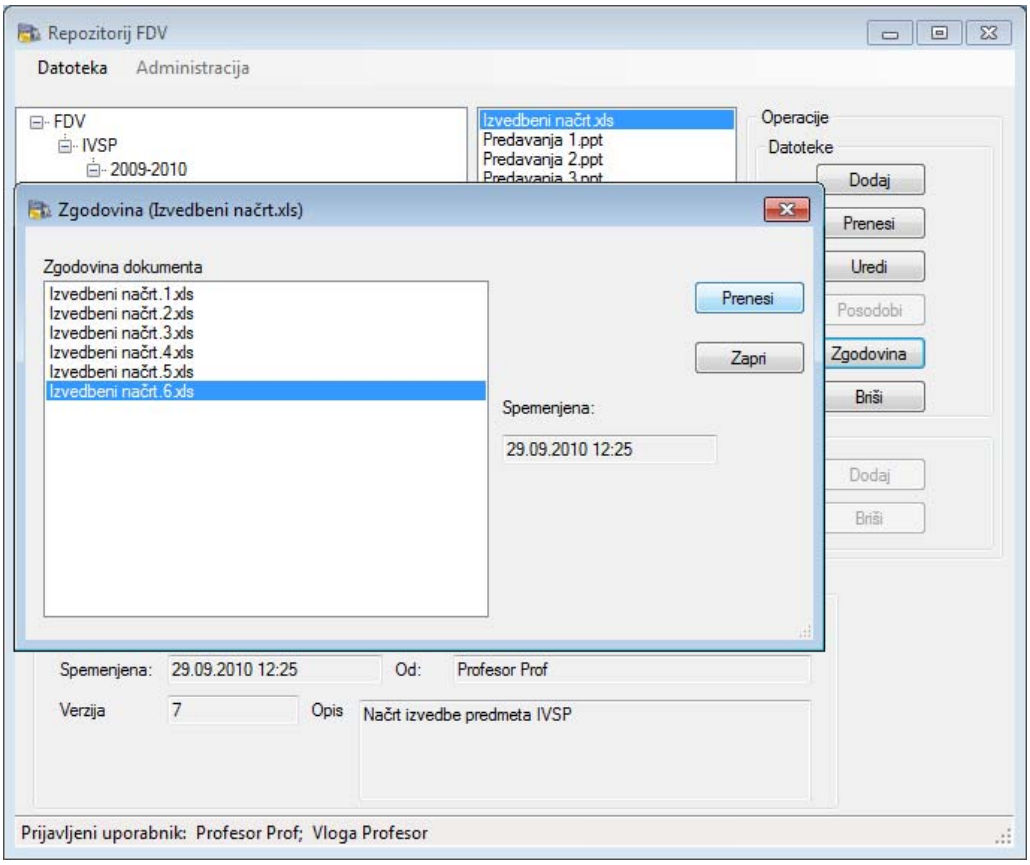
Gumb *Posodobi* se aktivira pri dokumentu, ki ga je prijavljen uporabnik predhodno zaklenil z gumbom *Uredi*. Kot pri dodajanju se tudi tokrat odpre novo okno, kjer uporabnik na svojem računalniku izbere spremenjeni dokument, ki ga želi posodobiti. Preden se ta proces zaključi, se trenutna izvedba v repozitoriju preimenuje (prvotnemu imenu se doda številka različice), nato pa se v repozitorij skopira nova spremenjena izvedba, v metapodatkih se spremeni številka različice in se odstrani oznaka *Zaklenjen* (glej prilogo C). Za ohranjanje vseh izvedb posameznega dokumenta se dokumenta ne sme preimenovati. Ko uporabnik želi posodobiti dokument, mora lta biti na računalniku shranjen pod istim imenom kot v repozitoriju. V nasprotnem primeru posodobitev ne bo uspela, v repozitorij pa se bo dodal kot nov dokument. Posodobitev bo uspela šele, ko bo dokument pravilno poimenovan, takrat se bo odstranila tudi oznaka *Zaklenjen*.

V prilogi C so deli izvorne kode, ki poskrbijo za izvedbo funkcij, ki jih omogočajo gumbi *Dodaj*, *Uredi* in *Posodobi*.

Gumb *Zgodovina* se aktivira pri dokumentu, nad katerim ima prijavljen uporabnik pravico pisanja (urejanje). Odpre se okno s seznamom vseh izvedb izbranega dokumenta (glej sliko 4.6), kjer lahko uporabnik izbere katero koli izvedbo dokumenta

in si jo shrani na računalnik. S klikom na posamezno izvedbo se v dodatnem polju izpišeta čas in datum njene spremembe.

Slika 4.6: Okno *Zgodovina*, kjer je seznam vseh izvedb izbranega dokumenta



Gumb *Briši* se aktivira, če ima uporabnik nad izbranim dokumentom pravico pisanja (urejanje). Aplikacija dokument izbriše iz repozitorija, XML-datoteka z metapodatki in prejšnje izvedbe pa se ohranijo na centralnem strežniku repozitorija.

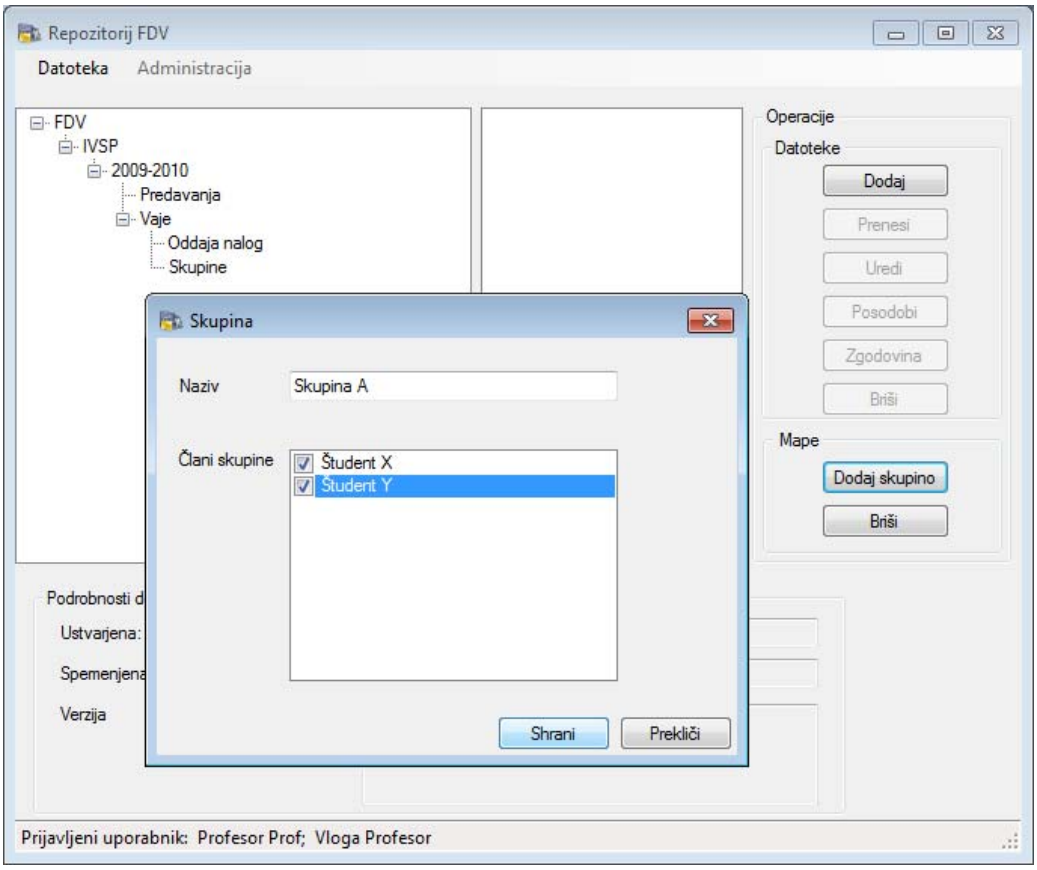
Operacije, ki se izvajajo nad mapami

Gumba *Dodaj* in *Briši* se aktivirata v mapah, kjer ima uporabnik z vlogo dodeljeno pravico dodajanja/brisanja map. S klikom na gumb *Dodaj* se odpre okno, kjer določimo ime nove mape, v repozitoriju se nato ustvari nova mapa in istoimenska skrita XML-datoteka z metapodatki.

Izjema je mapa *Skupine*, kjer se aktivira gumb *Dodaj skupino*, le če ima prijavljen uporabnik vlogo *Profesor*. Odpre se novo okno, kjer se določi naziv skupine in izbere njene člane, kot to prikazuje slika 4.7. Na seznamu so samo uporabniki z vlogo *Študent*, ki še niso bili predhodno vključeni v katero koli drugo skupino. Ko uporabnika z vlogo

Študent enkrat določimo neki skupini, se to namreč zabeleži med metapodatke tega uporabnika.

Slika 4.7: Okno za dodajanje podmap v mapo *Skupine*



Ko se postopek zaključi, se v repozitoriju ustvari mapa z določenim nazivom, med metapodatke pa se zabeleži, kdo ima pravico branja in pisanja v tej mapi. To so izbrani uporabniki z vlogo *Študent* in vsi uporabniki z vlogo *Profesor*.

Gumb *Briši* mapo izbriše iz repozitorija, prav tako izbriše tudi vse dokumente, ki se nahajajo v tej mapi. Pri brisanju mape znotraj mape *Skupine* se pri uporabnikih z vlogo *Študent*, ki so imeli pravico branja in pisanja v tej mapi, metapodatki spremenijo tako, da slednji niso več člani skupine.

4.2 Evalvacija vzorčne aplikacije

Pri razvoju aplikacije vedno obstaja več možnosti, kako naj se izvajajo posamezne funkcionalnosti. To je odvisno od vsakega programerja posebej in njegovega znanja ter načina razmišljanja. Tudi pri vzorčni aplikaciji je še veliko prostora za izboljšave, saj

ima določene pomanjkljivosti, zato bomo v tem podpoglavju ocenili izdelavo aplikacije in njeno smotrnost.

Vzorčna aplikacija je precej specifična in je ni možno uporabiti za drugačne namene. Izdelana je bila za točno določen problem. Pravice niso definirane le z vlogo uporabnika, temveč tudi s specifičnimi mapami, ki so že vnaprej določene, zato bi bilo treba narediti veliko sprememb, da bi se jo lahko uporabljalo širše in bolj množično. Ena od možnih sprememb za širšo uporabo je, da bi se pravice definirale le z vlogami. To bi bilo precej lažje, če bi aplikacija temeljila na relacijski bazi, saj se trenutno vsi podatki beležijo v XML-datoteke. Z bazo bi lahko drugače izvedli tudi katero drugo funkcionalnost, npr. verzioniranje, evidentiranje uporabnikov, beleženje metapodatkov ... Hkrati pa bi vključitev takšne baze omogočila tudi dodatne funkcionalnosti. Ena izmed njih je iskanje in poizvedbe po metapodatkih, kot so naziv datoteke, avtor, ključne besede …, vendar pa vključitev baze pomeni večjo kompleksnost aplikacije in posledično tudi več časa za njeno načrtovanje in izdelavo, kar presega meje zadanega problema (vzorčna aplikacija oz. prototip repozitorija).

Kot pomanjkljivost želimo izpostaviti tudi verzioniranje, ki bi lahko bilo bolj racionalno in ekonomično, kajti trenutno se vsakič, ko se posodobi nek dokument, celoten dokument skopira v repozitorij, kar je precej neekonomično glede izrabe prostora v repozitoriju. Boljša rešitev bi bila, če bi namesto celotnega dokumenta beležili le spremembe, ki jih uporabnik naredi v dokumentu, kot to dela odprtokodni sistem za nadzor izvedb *Subversion*.

Izpostavili smo le tiste pomanjkljivosti aplikacije, ki se nam zdijo najbolj pomembne, vendar bi nekdo, ki se tudi profesionalno ukvarja z razvojem aplikacij, izpostavil še katere druge napake, saj bi na problem lahko gledal iz čisto druge perspektive. Poudariti želimo, da je aplikacija le vzorčna, zato se ne more primerjati z drugimi že uveljavljenimi rešitvami, ki so na trgu že vrsto let.

5 Sklepne misli

S pomočjo računalnika in ustrezne programske opreme obdelujemo raznovrstne dokumente, npr. besedilo, slike, podatke, izvorno kodo ... Podatki, ki so del nekega dokumenta, so shranjeni v datoteki. Pogosto se zgodi, da dokument ne nastane naenkrat, ampak nastaja postopoma s spremembami oz. dopolnitvami, hkrati pa ga lahko ustvarja tudi več avtorjev. Če se celotnega procesa nastajanja dokumenta ne nadzoruje disciplinirano, slednje skoraj vedno povzroči nezaželeno izgubo podatkov. Uporaba repozitorijev, ki skrbijo za ohranjanje zgodovine nastanka dokumentov ter zaščito pred nezaželeno izgubo podatkov, je zato dobrodošla na vseh področjih, kjer se uporablja informacijsko-komunikacijska tehnologija.

V diplomskem delu je predstavljen pristop izdelave lastnega repozitorija za specifičen problem. Gre za prototip repozitorija oz. vzorčno aplikacijo, ki predavateljem in študentom pri predmetu *Informacijski viri in sekundarni podatki* omogoča shranjevanje gradiv, potrebnih za ta predmet na enem mestu. Ta gradiva lahko obnavljajo, pri tem se ta ustrezno verzionirajo, njihove starejše različice pa se ohranijo in se do njih lahko kadar koli dostopa. Odločili smo se, da repozitorij sprogramiramo sami, kar ni pogost pristop. V našem primeru je bila to boljša odločitev, saj smo lahko upoštevali vse želje in zahteve, hkrati pa smo lahko preučili, kako repozitorij deluje v osnovi. To pa ni vedno najboljša rešitev, predvsem če smo časovno in finančno omejeni, saj je na voljo veliko programskih rešitev – tako brezplačnih kot tudi plačljivih, ki nudijo nekatere osnovne funkcionalnosti repozitorijev.

Pri koncu pisanja diplomskega dela smo se seznanili še z enim primerom repozitorija, to je storitev za gostovanje in izmenjavo datotek na internetu *Dropbox*, ki ga je tudi smiselno preučiti v primeru odločanja o uporabi ali razvoju rezpozitorija (Wikipedia 2008). Slednji je uporabnikom bolj prijazen za uporabo kot *Subversion*, hkrati pa precej podoben našemu prototipu, le da ni tako specifično usmerjen (na pravice in vloge).

Pred izdelavo lastnega repozitorija je pomembno, da se najprej dobro opredeli problem in potrebe, razišče obstoječe rešitve in šele nato sprejme res z informacijami podprto odločitev, ali uporabiti eno od že obstoječih rešitev ali razviti lastno programsko rešitev.

V času pisanja diplomskega dela smo pridobili veliko novega znanja, spoznali smo tako delovanje repozitorijev kot tudi osnovne koncepte načrtovanja in razvoja aplikacij.

6 Literatura

1. Ambrožič, Melita, Mojca Šavnik, Zoran Krstulović, Uroš Katić in Špela Svoljšak. 2006. *Strategija razvoja Digitalne knjižnice Slovenije – dLib.si 2007-2010.* Dostopno prek: http://www.dlib.si/v2/documents/pdf/Strategija_Dlib.pdf (3. april 2010).
2. Božeglav, Domen. 2009. *Storitve izobraževalne in raziskovalne mreže za srednje šole.* Magistrska naloga. Dostopno prek: http://www.ediplome.fm-kp.si/Bozeglav_Domen_20091214.pdf (8. april 2010).
3. Collins - Sussman, Brian W. Fitzpatrick in C. Michael Pilato. 2002–2005. *Version Control with Subversion – For Subversion 1.1*. Dostopno prek: http://svnbook.red-bean.com/en/1.1/index.html (28. september 2010).
4. ELefANTC. 2010. *E-učenje in izobraževanje na daljavo.* Dostopno prek: http://elefantc.sparc.uni-mb.si/eUcenje.html (3. april 2010).
5. Fakulteta za družbene vede. 2010. *Predmeti in predmetniki: Informacijski viri in sekundarni podatki.* Dostopno prek: https://prijava.fdv.uni-lj.si/javno/Predmeti_3.asp?id_nacrt=1063 (15. september 2010).
6. Gabrijelčič, Primož. 2010. Različice dokumentov. *Revija Monitor* (Marec). Dostopno prek: http://www.monitor.si/clanek/razlicice-dokumentov/ (15. september 2010).
7. Heery, Rachel in Sheila Anderson. 2005. *Digital Repositories Review*. Joint Information Systems Committee. Dostopno prek: http://www.jisc.ac.uk/uploaded_documents/digital-repositories-review-2005.pdf (28. marec 2010).
8. JISC infoNet. 2011. *Introduction to the Digital Repositories infoKit.* Dostopno prek: http://www.jiscinfonet.ac.uk/infokits/repositories/ (28. april 2011).
9. Kavčič - Čolić, Alenka in Mateja Šmid. 2007. Evalvacija zaupanja vrednih digitalnih arhivov pri projektu reUSE!. *Knjižnica* 51 (1): 115–140.
10. Küng, Stefan, Lübbe Onken in Simon Large, prevod Matjaž Čepon. 2010. *TortoiseSVN: Odjemalec za Subversion v operacijskem sistemu Windows – version 1.6*. Dostopno prek: http://tortoisesvn.net/docs/nightly/TortoiseSVN_sl/index.html (29. september 2010).
11. Lokar, Matija in Srečo Uranič. 2008. *Programski jezik C# – osnovni koraki*. Dostopno prek: http://arh.tsckr.si/IOD/Literatura/C%23_UVOD.pdf (30. september 2010).

12. Marjetica.net. 2010. *Model sistema za ravnateljevanje z vsebino.* Dostopno prek: http://www.marjetica.net/marjetica/page.php?block=sl_cms_model (3. april 2010).
13. Mihelič, Jurij. 2010. *Subversion.* Dostopno prek: http://lalg.fri.uni-lj.si/~jure/doku.php?id=subversion (16. september 2010).
14. Repositories Support Project. 2010a. *Open access?* Dostopno prek: http://www.rsp.ac.uk/start/before-you-start/open-access/ (28. marec 2010).
15. --- 2010b. *Technical approaches.* Dostopno prek: http://www.rsp.ac.uk/start/setting-up-a-repository/technical-approaches/ (29. marec 2010).
16. --- 2010c. *What is a repository?* Dostopno prek: http://www.rsp.ac.uk/start/before-you-start/what-is-a-repository/ (28. marec 2010).
17. Šverko, Peter. 2006. *Upravljanje elektronskih dokumentov.* Diplomsko delo. Dostopno prek: http://www.cek.ef.uni-lj.si/u_diplome/sverko2295.pdf (3. april 2010).
18. Vipavc, Irena in Jožica Klep. 2003. Uskladitev metapodatkovnih standardov za potrebe uporabnikov: primer METIS in DDI. V *Statistika kot orodje in vir za kreiranje znanja uporabnikov*, ur. Boris Tkačik, 535–544. Ljubljana: Statistični urad Republike Slovenije, Statistično društvo Slovenije.
19. Wikipedia. 2005. *Lock (database)*. Dostopno prek: http://en.wikipedia.org/wiki/Lock_(database) (30. marec 2010).
20. --- 2006. *Open Archives Initiative Protocol for Metadata Harvesting.* Dostopno prek: http://en.wikipedia.org/wiki/Open_Archives_Initiative_Protocol_for_Metadata_Harvesting (22. marec 2010).
21. --- 2007. *Software repository*. Dostopno prek: http://en.wikipedia.org/wiki/Software_repository (29. marec 2010).
22. --- 2008. *Dropbox (service)*. Dostopno prek: http://en.wikipedia.org/wiki/Dropbox_(service) (15. maj 2011).

Priloge

Priloga A: Primer XML-datoteke za dodajanje uporabnikov

Tukaj je primer vsebine XML-datoteke, ki se lahko uporabi pri dodajanju večjega števila novih uporabnikov.

```
<?xml version="1.0" encoding="utf-8"?>
<Uporabniki xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
	<Uporabnik>
		<Ime>Petra</Ime>
		<Priimek>Cimerman</Priimek>
		<UporabniskoIme>pcim</UporabniskoIme>
		<Geslo>petra</Geslo>
		<VlogaId>3</VlogaId>
	</Uporabnik>
	<Uporabnik>
		<Ime>Tone</Ime>
		<Priimek>Novak</Priimek>
		<UporabniskoIme>tnov</UporabniskoIme>
		<Geslo>tone</Geslo>
		<VlogaId>3</VlogaId>
	</Uporabnik>
	<Uporabnik>
		<Ime>Peter</Ime>
		<Priimek>Klepec</Priimek>
		<UporabniskoIme>pkle</UporabniskoIme>
		<Geslo>peter</Geslo>
		<VlogaId>3</VlogaId>
	</Uporabnik>
	<Uporabnik>
		<Ime>Luka</Ime>
		<Priimek>Kralj</Priimek>
		<UporabniskoIme>lkra</UporabniskoIme>
		<Geslo>luka</Geslo>
		<VlogaId>3</VlogaId>
	</Uporabnik>
	<Uporabnik>
		<Ime>Janja</Ime>
		<Priimek>Gorc</Priimek>
		<UporabniskoIme>jgor</UporabniskoIme>
		<Geslo>janja</Geslo>
		<VlogaId>3</VlogaId>
	</Uporabnik>
	<Uporabnik>
		<Ime>Anja</Ime>
		<Priimek>Pepelka</Priimek>
		<UporabniskoIme>apep</UporabniskoIme>
		<Geslo>anja</Geslo>
		<VlogaId>3</VlogaId>
	</Uporabnik>
</Uporabniki>
```

Priloga B: Predstavitev izbranih metod v določenih razredih

Tukaj je predstavljenih nekaj izbranih metod v določenih razredih, ki tvorijo vzorčno aplikacijo.

Razred *RepoManager:* skrbi za delovanje repozitorija in nam omogoča, da repozitorij upravljamo preko naslednjih metod:

- *CreateRepository*: na lokalnem disku ob prvem zagonu ustvari repozitorij s privzeto strukturo map in doda privzete pravice.
- *CreateDirectory*: ustvari mapo v repozitoriju.
- *CreateDirectoryForStudent*: ustvari mapo za študenta.
- *CreateDirectoryForSkupina*: ustvari mapo za skupino.
- *OdstraniSkupino*: zbriše mapo za skupino iz repozitorija in uporabniku v metapodatkih spremeni parametre, da ne pripada več nobeni skupini.
- *DeleteDirectory*: zbriše mapo iz repozitorija.
- *SaveFile*: shrani dokument in ga ustrezno verzionira.
- *LockFile*: zaklene izbrano datoteko, če je možno.
- *IsFileLocked*: preveri, ali je dokument že zaklenjen in od koga.
- *CanReadFile*: za izbrano datoteko pove, ali ima prijavljen uporabnik pravico branja.
- *CanWriteFile*: za izbrano datoteko pove, ali ima prijavljen uporabnik pravico pisanja.
- *GetFolderMetaData*: vrne metapodatke o mapi.
- *GetFileMetaData*: vrne metapodatke o datoteki.

Razred *Security:* skrbi za prijavo in avtorizacija uporabnikov.

- *Login*: preveri uporabniško ime in geslo med registriranimi uporabniki.
- *IsInRole*: preveri, ali ima prijavljen uporabnik vlogo, ki je zahtevana za določeno akcijo.

Razred *MainForm:* odpre glavno okno in skrbi, da se odprejo vsa ostala glede na operacijo.

- *ConfigurateButtons*: določa, kdaj se aktivirajo določeni gumbi.

Razred *Uporabnik*: predstavlja uporabnika v našem programu.

- *New*: dodajanje novega uporabnika
- *TryGetById*: poizvedba po uporabniku glede na njegov ID.
- *SaveForSkupina*: uporabnika določi skupini in prepreči, da se ne pojavi v več skupinah.
- *SaveFromXML*: uvoz uporabnikov iz vnaprej pripravljenega XML-ja v aplikacijo.

Razred *FileMetaData*: vsebuje metapodatke o tem, kakšne pravice ima prijavljeni uporabnik nad določenim dokumentom.

Razred *FolderMetaData*: vsebuje metapodatke o tem, kakšne pravice ima prijavljeni uporabnik v določeni mapi.

Razred *Vloga*: definira vloge uporabnikov v programu.

Razred *XmlHelper*: skrbi za upravljanje z XML-datotekami. Shrani podatke v določeno XML-datoteko (*SaveToXml*) in prebere podatke v določeni XML-datoteki (*LoadFromXml*).

Priloga C: Posamezni deli izvorne kode vzorčne aplikacije

Na naslednjih straneh se nahaja izvorna koda za dodajanje dokumenta, dodajanje pravic, urejanje (zaklepanje) in posodabljanje (verzioniranje) ter dodajanje map v *Skupine*.

Dodajanje dokumenta (v primeru posodobitve je verzioniranje)

```
class RepoManager
/// <summary>
/// Shrani dokument.
/// Če dokument v repozitoriju ne obstaja, naredi novega z različico
1.
/// Če obstaja pa ga ustrezno verzionira.
/// </summary>
/// <param name="sourceFilePath">Izvorna pot dokumenta</param>
/// <param name="destFilePath">Ciljna pot dokumenta</param>
/// <param name="metaData">Metapodatki</param>
/// <param name="vrstaMape">Vrsta mape kamor se bo dokument
dodal</param>
public void SaveFile(string sourceFilePath, string destFilePath,
      FileMetaData metaData, VrstaMape vrstaMape)
{
      if (!File.Exists(destFilePath))
      {
            metaData.Verzija = 1;
            string xmlName = Path.ChangeExtension(destFilePath,
".xml");

            // Skopira datoteko v repozitorij.
            File.Copy(sourceFilePath, destFilePath);

            // Nastavi avtorja in zabeleži datum kreacije in spremembe.
            metaData.Avtor = metaData.ChangedBy =
                  Security.PrijavljeniUporabnik.Id;
            metaData.ChangedOn = metaData.CreatedOn = DateTime.Now;
            metaData.DodajPravice(vrstaMape);

            // Shrani metapodatke.
            XmlHelper.SaveToXml<FileMetaData>(metaData, xmlName);
            File.SetAttributes(xmlName, FileAttributes.Hidden);
       }
       else
       {
            // Naloži metapodatke dokumenta.
            metaData = XmlHelper.LoadFromXml<FileMetaData>(
                  Path.ChangeExtension(destFilePath, ".xml"));

            // Preveri, če je dokument zaklenjen od trenutnega
uporabnika.
            if (metaData.Lock && metaData.LockOwner ==
                  Security.PrijavljeniUporabnik.Id)
            {
                // Preimenuje staro različico in jo shrani.
                string koncnica = Path.GetExtension(destFilePath);
                string arhivFileName = Path.ChangeExtension(
                        destFilePath, metaData.Verzija + koncnica);
                File.Copy(destFilePath, arhivFileName);
                File.SetAttributes(arhivFileName,
FileAttributes.Hidden);
```

```
// Verzijo poveča, odstrani ključavnico ter zabeleži datum in avtorja
spremembe.
                metaData.Verzija += 1;
                metaData.Lock = false;
                metaData.LockOwner = 0;
                metaData.ChangedOn = DateTime.Now;
                metaData.ChangedBy = Security.PrijavljeniUporabnik.Id;
                //Skopira dokument v repozitorij in shrani metapodatke.
                File.Copy(sourceFilePath, destFilePath, true);
                XmlHelper.SaveToXml<FileMetaData>(
                   metaData, Path.ChangeExtension(destFilePath,
                  ".xml"));
             }
         }
}
```

Glede na vlogo in mapo se daje pravico branja in pisanja

```
class FileMetaData
/// <summary>
/// Doda pravice na dokument, glede na vrsto mape in vlogo.
/// </summary>
/// <param name="vrstaMape">Vrsta mape</param>
internal void DodajPravice(VrstaMape vrstaMape)
{
      List<int> read = new List<int>();
      List<int> write = new List<int>();
      Vloga vloga = Vloga.List.Single(v => v.Id ==
      Security.PrijavljeniUporabnik.VlogaId);
      read.Add(Vloga.Administrator.Id);
      write.Add(Vloga.Administrator.Id);

      if ((vrstaMape == VrstaMape.Predavanja) ||
         (vrstaMape == VrstaMape.Skupine) || (vrstaMape ==
      VrstaMape.Vaje))
      {
            if (vloga == Vloga.Profesor)
            {
                read.Add(Vloga.Profesor.Id);
                read.Add(Vloga.Student.Id);

                write.Add(Vloga.Profesor.Id);
            }
      }
      else if (vrstaMape == VrstaMape.Student)
      {
            if (vloga == Vloga.Student)
            {
                read.Add(Vloga.Student.Id);
                 write.Add(Vloga.Student.Id);
            }
      }
      else if (vrstaMape == VrstaMape.OddajaNalog)
      {
            read.Add(Vloga.Profesor.Id);
            if (vloga == Vloga.Profesor)
            {
                write.Add(Vloga.Profesor.Id);
                read.Add(Vloga.Student.Id);
            }
       }
```

```
        else if (vrstaMape == VrstaMape.Skupina)
        {
             if (vloga == Vloga.Student)
             {
                  read.Add(Vloga.Student.Id);
                  write.Add(Vloga.Student.Id);
                  read.Add(Vloga.Profesor.Id);
             }
             else if(vloga == Vloga.Profesor)
             {
                  read.Add(Vloga.Student.Id);
                  write.Add(Vloga.Profesor.Id);
                  read.Add(Vloga.Profesor.Id);
             }
         }
         Write = write.ToArray();
         Read = read.ToArray();
}
```

Uredi (zaklepanje dokumenta):

```
partial class MainForm: Form
/// <summary>
/// Ob kliku na gumb Uredi se dokument shrani in zaklene.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnUredi_Click(object sender, EventArgs e)
{
    int? lockOwner;
    // Preveri, če je dokument zaklenjen.
    if (!_repoManager.IsFileLocked(CurrentPath, out lockOwner))
    {
         SaveFileDialog savefileDialog = new SaveFileDialog();
         savefileDialog.FileName = _currentSelectedFile;
         if (savefileDialog.ShowDialog() == DialogResult.OK)
         {
              // Prekopira iz repozitorija datoteko na izbrano mesto.
              File.Copy(CurrentPath, savefileDialog.FileName, true);

              // Zaklene datoteko v repozitoriju.
              _repoManager.LockFile(CurrentPath);

              // Ustrezno omogoči/onemogoči gumbe.
              ConfigurateButtons();
         }
     }
     else
     {
          // Opozori uporabnika.
          MessageBox.Show(string.Format("Datoteka {0} je zaklenjena od
      uporabnika - {1}",
          Path.GetFileName(CurrentPath),
Uporabnik.TryGetById(lockOwner)));
      }
}
class RepoManager
/// <summary>
/// Zaklene izbrani dokument.
/// </summary>
// <param name="filePath">Pot do dokumenta</param>
```

```
/// <returns>Vrne true, če je uspel zakleniti dokument, drugače
false</returns>
public bool LockFile(string filePath)
{
     // Pridobi metapodatke.
     FileMetaData metaData = XmlHelper.LoadFromXml<FileMetaData>(
         Path.ChangeExtension(filePath, ".xml"));
     // Če je dokument zakljenjen, vrne false.
     if (metaData.Lock)
         return false;

     // Zaklene dokumet in shrani metapodatke.
     metaData.Lock = true;
     metaData.LockOwner = Security.PrijavljeniUporabnik.Id;
     XmlHelper.SaveToXml<FileMetaData>(
         metaData, Path.ChangeExtension(filePath, ".xml"));

     return true;
}
```

Posodobi:

```
class MainForm
/// <summary>
/// Ob kliku na gumb Posodobi se izvede ta metoda.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnPosodobi_Click(object sender, EventArgs e)
{
// Preveri, če je dokument zaklenjen s strani prijavljenega
uporabnika.
    if (_repoManager.IsFileLockedByUser(
      CurrentPath, Security.PrijavljeniUporabnik))
    {
         // Doda dokument v repozitorij.
         DodajVRepozitorij(false);

         // Osveži prikaz dokumentov.
         DisplayFiles(_currentSelectedFolder);
     }
}
```

```
class MainForm
/// <summary>
/// Odpre okno za dodajanje dokumentov, kjer uporabnik izbere dokument
in shrani dokument v repozitorij.
/// </summary>
/// <param name="novDokument">Je nov ali že obstoječi dokument</param>
private void DodajVRepozitorij(bool novDokument)
{
     DodajDokumentForm form = new DodajDokumentForm(novDokument);

     // Če uporabnik klikne gumb Shrani.
     if (UIController.OpenFormInDialog(form) == DialogResult.OK)
     {
          string destFileName = _currentSelectedFolder + @"\" +
              Path.GetFileName(form.FilePath);
          // Shrani dokument.
          _repoManager.SaveFile(form.FilePath, destFileName,
              form.DataSource, CurrentFolderMetaData.Vrsta);
```

```
    }
}
```

Dodajanje mape za skupino:

```
public class RepoManager
/// <summary>
/// Ustvari mapo za skupino.
/// </summary>
/// <param name="dirPath">Lokacija mape</param>
/// <param name="nazivMape">Naziv mape</param>
/// <param name="claniSkupine">Člani skupine</param>
public void CreateDirectoryForSkupina(string dirPath, string
nazivMape, List<Uporabnik> claniSkupine)
{
      string newDirPath = dirPath + @"\" + nazivMape;
      Directory.CreateDirectory(newDirPath);

      FolderMetaData metaData = new FolderMetaData(VrstaMape.Skupina);
      metaData.Avtor = Security.PrijavljeniUporabnik.Id;
      metaData.ReadUsers = claniSkupine.Select(u => u.Id).ToArray();
      metaData.WriteUsers = claniSkupine.Select(u => u.Id).ToArray();

      // Doda pravice za administratorja in profesorja.
      metaData.ReadVloge = new int[] { Vloga.Administrator.Id,
                                       Vloga.Profesor.Id };
      metaData.WriteVloge = new int[] {  Vloga.Profesor.Id };

      string xmlName = newDirPath + @"\" + nazivMape + ".xml";

      // Kreira xml za metapodatke.
      XmlHelper.SaveToXml<FolderMetaData>(metaData, xmlName);
      File.SetAttributes(xmlName, FileAttributes.Hidden);

      // Posodobi uporabnike.
      Uporabnik.SaveForSkupina(claniSkupine, true);
}
public class Uporabnik
/// <summary>
/// Posodobi uporabnike, ki so dodani ali odstranjeni iz skupine.
/// </summary>
/// <param name="list">Člani skupine</param>
/// <param name="create">True - dodan, false - odstranjen</param>
public static void SaveForSkupina(List<Uporabnik> list, bool create)
{
   List<Uporabnik> upoList = Uporabnik.GetAll();

   foreach (var uporabnik in list)
   {
      upoList.Single(upo => (upo.Id == uporabnik.Id && upo.VlogaId ==
                             Vloga.Student.Id)).JeVSkupini = create;
   }

   Uporabniki uporabniki = new Uporabniki();
   foreach (var upo in upoList)
   {
      uporabniki.Add(upo);
   }
   XmlHelper.SaveToXml<Uporabniki>(uporabniki,
Config.UporabnikDataPath);
}
```

Priloga Č: Vzorčna aplikacija RepozitorijFDV

Priloga Č je celotna naša vzorčna aplikacija *RepozitorijFDV*, ki je priložena na zgoščenki skupaj z diplomsko nalogo v elektronski različici; nahaja se v mapi *Priloga Č*. Ta vsebuje mapo *RepoBrowser* z datotekami izvorne kode in datoteke, ki omogočajo delovanje aplikacije, med katerimi je ključna datoteka *RepozitorijFDV.exe*. Z dvojnim klikom nanjo se zažene aplikacijo.

Za delovanje aplikacije je treba imeti na računalniku naložen operacijski sistem *Windows* in že omenjeni *.NET Framework 3.5 SP1*. Program je brezplačen, dostopen je na spletni strani http://www.microsoft.com/downloads/en/details.aspx?FamilyID=ab99342f-5d1a-413d-8319-81da479ab0d7. V primeru operacijskega sistema *Windows 7* namestitev tega programa ni potrebna.