Univerzitet u Beogradu
Matematički fakultet

Mirjana Maljković

Dizajn i implementacija podsistema za nastavničke beleške sistema STUDINFO

master rad

Beograd
2010.

Univerzitet u Beogradu – Matematički fakultet
Master rad

Autor: Mirjana Maljković

Mentor: dr Saša Malkov
Univerzitet u Beogradu – Matematički fakultet

Članovi komisije: dr Nenad Mitić
Univerzitet u Beogradu – Matematički fakultet

dr Gordana Pavlović-Lažetić
Univerzitet u Beogradu – Matematički fakultet

Datum odbrane: 23. septembar 2010.

Sadržaj

1 Uvod

Informacioni sistem StudInfo predstavlja integrisani sistem koji omogućava redovan rad studentske službe, podršku nastavnog procesa i praćenje studentskih aktivnosti i upotrebljava se na više fakulteta Univerziteta u Beogradu. Informacioni sistem čine tri osnovne aplikativne celine namenjene studentskoj službi, nastavnicima i studentima.

Informacioni sistem StudInfo razvija odeljenje za razvoj softvera u okviru Računarske labaratorije na Matematičkom fakultetu sa ciljem da se obezbedi informatička podrška implementaciji Zakona o visokom obrazovanju Republike Srbije od 2.9.2005. godine.

Aplikativna celina namenjena nastavnicima podržava nastavne i ispitne aktivnosti tako što omogućava praćenje ispunjavanja studentskih predispitnih i ispitnih obaveza. Nastavnicima i saradnicima je omogućeno da za svaki pojedinačan kurs:

- održavaju spisak predviđenih obaveza;
- održavaju podatke o ispunjavanju obaveza;
- objavljuju obaveštenja za studente;
- unose rezultate ispita.

Tako se omogućava jednostavno, efikasno i pouzdano razmenjivanje informacija između nastavnika, studenata i studentske službe.

Kako bi se unapredilo i olakšalo vođenje nastavne aktivnosti, javlja se potreba za mogućnošću da nastavnici i saradnici beleže njima bitne podatke o kursevima i studentima. Unapređenje se može ostvariti uvođenjem podsistema za beleške u okviru postojećeg sistema koji bi omogućio da nastavnik ili saradnik beleži podatke o svim kursevima i studentima u čijoj nastavi učestvuje u tekućoj školskoj godini. Podsistem za vođenje beleški bi olakšao i komunikaciju između nastavnika koji učestvuju u održavanju nastave iz određenog kursa ili predaju istom student ako bi se omogućilo i deljenje beleški.

U radu su opisani projektovanje i implementacija podsistema za vođenje nastavničkih beleški koji je razvijen kao deo informacionog sistema StudInfo. Pri projektovanju i implementaciji upotrebljene su različite tehnike objektno-orijentisanog projektovanja i razvoja softvera.

1.1 Metodologija analize i projektovanja

Pri analizi je primenjena tehnika dijagrama toka podataka (eng. *Data Flow Diagram – DFD)* da bi se na formalan način modeliralo funkcionisanje podsistema i ilustrovale aktivnosti koje se obavljaju, kao i način kretanja podataka između njih.

U analizi su korišćeni slučajevi upotrebe da bi se opisalo željeno ponašanje podsistema za nastavničke beleške sa aspekta korisnika sistema. Na osnovu slučajeva upotrebe projektovan je korisnički interfejs.

Za projektovanje baze podataka korišćen je model entiteta i odnosa i dijagramska tehnika za prikaz entiteta i odnosa među njima, zajedno sa njihovim karakteristikama. Shema dobijena primenom modela entiteta i odnosa preslikana je u relacionu bazu podataka i korišćena kao osnova za definisanje klasa podataka prilikom implementacije podsistema za nastavničke beleške.

Dijagrami toka podataka

Dijagram toka podataka grafički prikazuje tok podataka između spoljašnjih entita, procesa u sistemu i struktura za skladištenje podataka. Služi za okvirno razumevanje dešavanja u sistemu, bez detaljnog razmatranja konkretnih procesa. Elementi dijagrama toka podataka su:

- proces - aktivnost ili funkcija koja se obavlja iz određenog poslovnog razloga. Grafički se prikazuje simbolom:

Slika 1: Grafički prikaz procesa u dijagramu toka podataka

- spoljašnji entitet – predstavlja osobu, organizaciju ili drugi sistem koji interaguje sa sistemom koji se modelira. Grafički se prikazuje simbolom:

Slika 2: Grafički prikaz entiteta u dijagramu toka podataka

- tok podataka - podatak ili logički skup podataka koji polazi iz ili završava u procesu. Prikazuje se strelicom od spoljašnjeg entiteta ili skladišta podataka ka procesu.

- skladište podataka - skup podataka koji se čuvaju na neki način. Ako podatak izlazi iz skladišta onda se upotrebljava, a ako ulazi u skladište onda se ažurira postojeći ili se dodaje novi podatak. Skladište podataka grafički se prikazuje simbolom:

Slika 3: Grafički prikaz skladišta podataka u dijagramu toka podataka

U radu je dijagram toka podataka za podsistem za nastavničke beleške prestavljen na tri nivoa:

- dijagram konteksta – ceo podsistem je predstavljen jednim procesom;
- dijagram prvog nivoa – uočavaju se glavni procesi u podsistemu;
- dijagram drugog nivoa – svaki proces iz dijagrama prvog nivoa predstavljen je jednim dijagramom na kome su detaljnije prikazani njegovi elementi.

Slučajevi upotrebe

Slučaj upotrebe (eng. *use case*) opisuje interakciju između učesnika i sistema i predstavlja opis niza akcija koje sistem izvodi. Jedan slučaj upotrebe odgovara jednoj funkciji sistema, čiji rezultat ima odgovarajuću vrednost za učesnika. Pomoću slučaja upotrebe se dokumentuje ponašanje sistema sa tačke gledišta učesnika. Slučaj upotrebe se grafički predstavlja elipsom i njegovo ime se upisuje u elipsu.

Slika 4: Grafički prikaz slučaja upotrebe

Slučajevi upotrebe se povezuju sa učesnicima (eng. *actor*). Učesnik je najčešće čovek, ali može biti hardverski uređaj ili čak drugi sistem. Predstavlja se grafičkim simbolom:

Slika 5: Grafički prikaz učesnika u dijagramu slučajeva upotrebe

Slučaj upotrebe se opisuje tekstom u vidu niza koraka. Ne postoji jedinstven propisan način kako se opisuje slučaj upotrebe, već postoje predložene stavke koje se mogu koristiti za opis svakog slučaja upotrebe. U ovom radu za slučaj upotrebe navode se:

- ime
- kratak opis
- učesnici
- tok događaja
  - osnovni tok
  - alternativni tokovi
- preduslovi
- postuslovi.

Dijagram slučajeva upotrebe predstavlja grafički prikaz slučajeva upotrebe. Na njemu su predstavljeni:

- učesnici;
- slučajevi upotrebe;
- asocijacije - odnosi između učesnika i slučajeva upotrebe. Predstavljaju se u obliku linije. Asocijacija između učesnika i slučaja upotrebe postoji ako je učesnik uključen u interakciju koja je opisana slučajem upotrebe;
- odnosi između slučajeva upotrebe.

Model entiteta i odnosa

U modelu entiteta i odnosa osnovni pojmovi su entitet, tip entiteta i odnos između tipova entiteta.

Entitetom se modelira osoba, mesto, pojam ili stvar o kojoj je potrebno voditi podatke. Svaki entitet se opisuje atributima koji su karakteristični za njega. Tip entiteta predstavlja skup entiteta sa istim skupom atributa.

Odnos predstavlja vezu između tipova entiteta. Svaki odnos ima kardinalnost, minimalan i maksiman broj entiteta jednog tipa koji se mogu pridružiti jednom entitetu drugog tipa. Postoje tri osnovne kardinalnosti binarnog odnosa (između tipova entiteta X i Y):

- 1:1 („jedan prema jedan“) - jedan entitet tipa X može biti pridružen najviše jednom entitetu tipa Y. Jedan entitet tipa Y može biti pridružen najviše jednom entitetu tipa X.
- 1:M („jedan prema više“) - jedan entitet tipa X može biti pridružen većem broju entiteta tipa Y. Jedan entitet tipa Y može biti pridružen najviše jednom entitetu tipa X.
- M:M („više prema više“) - jedan entitet tipa X može biti pridružen većem broju entiteta tipa Y. Jedan entitet tipa Y može biti pridružen većem broju entiteta tipa X.

Odnos M:M između dva tipa entiteta (ili više tipova entiteta) se opisuje posebnim tipom entiteta – asocijativnim entitetom. Acocijativni entitet može imati i svoja sopstvena svojstva.

1.2 Tehnologije primenjene pri implementaciji

Informacioni sistem StudInfo predstavlja veb aplikaciju kojoj korisnici pristupaju preko Interneta koristeći veb pretraživače. Na taj način je omogućen lak pristup aplikaciji bez potrebe za distribucijom i instalacijom dodatnog softvera i mogućnost njene upotrebe bez obzira na platformu i mesto pristupanja. Potrebno je samo da je računar sa kog želi da se pristupi veb aplikaciji povezan sa Internetom i ima veb pretraživač.

Kako podsistem za nastavničke beleške predstavlja proširenje informacionog sistema StudInfo, za njegovu izradu su korišćene tehnologije koje se primenjuju u izradi informacionog sistema StudInfo.

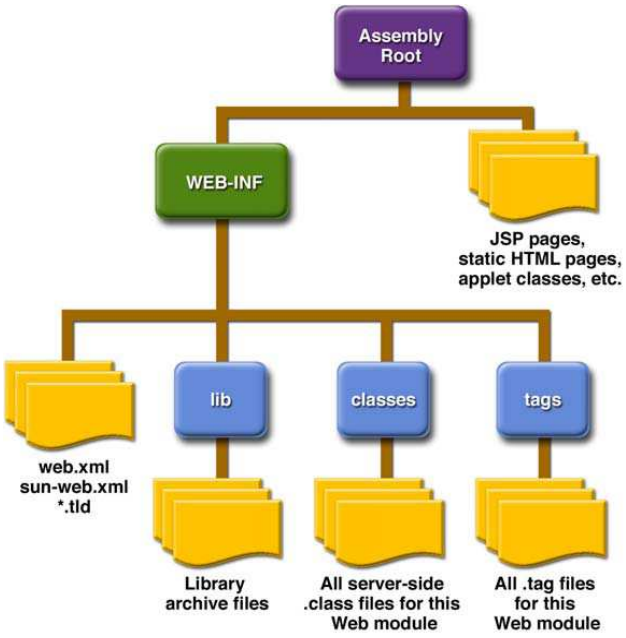
Veb aplikacija

Veb aplikacija je višekorisnička aplikacija koja kao korisnički interfejs upotrebljava veb tehnologije. Veb aplikacije se uobičajeno razvijaju primenom troslojne ili višeslojne arhitekture.

Arhitektura veb aplikacije informacionog sistema StudInfo je troslojna. Slojevi i tehnologije koje se koriste u njima su:

- prezentacioni sloj – koristi se veb pregledač (*MS Internet Explorer*, *Mozilla Firefox* ili dr.) i *JavaScript* za implementaciju aktivnih delova korisničkog interfejsa.
- aplikativni sloj – za implementaciju ponašanja sistema upotrebljen je programski jezik *Java*. Za povezivanje sa prezentacionim slojem koristi se veb/aplikativni server *Tomcat* i tehnologija *Java Server Pages*. Za povezivanje sa slojem podataka upotrebljen je *Hibernate*.
- sloj podataka – koristi se sistem za upravljanje bazom podataka *IBM DB2*.

Java veb aplikacije imaju propisanu strukturu direktorijuma. Osnovni direktorijum aplikacije naziva se *koreni direktorijum*. U korenom direktorijumu, ili njegovim poddirektorijumima, se nalaze statičke *HTML* stranice. Rezervisan direktorijum *WEB-INF* sadrži datoteku *web.xml*, koja se koristi za inicijalizaciju veb alplikacije. U direktorijumu *lib* se nalaze *jar* datoteke kojima se obezbeđuju nestandardne osobine veb aplikacije. U direktorijumu *class* se nalaze programi i pomoćne datoteke. Samo je koreni direktorijum neposredno vidljiv preko Interneta, zbog čega se *HTML* stranice nalaze u njemu. Direktorijumu *WEB-INF* i njegovom sadržaju se ne može neposredno pristupiti preko Interneta. Struktura direktorijuma *Java* veb aplikacije je prikazan na slici 6.



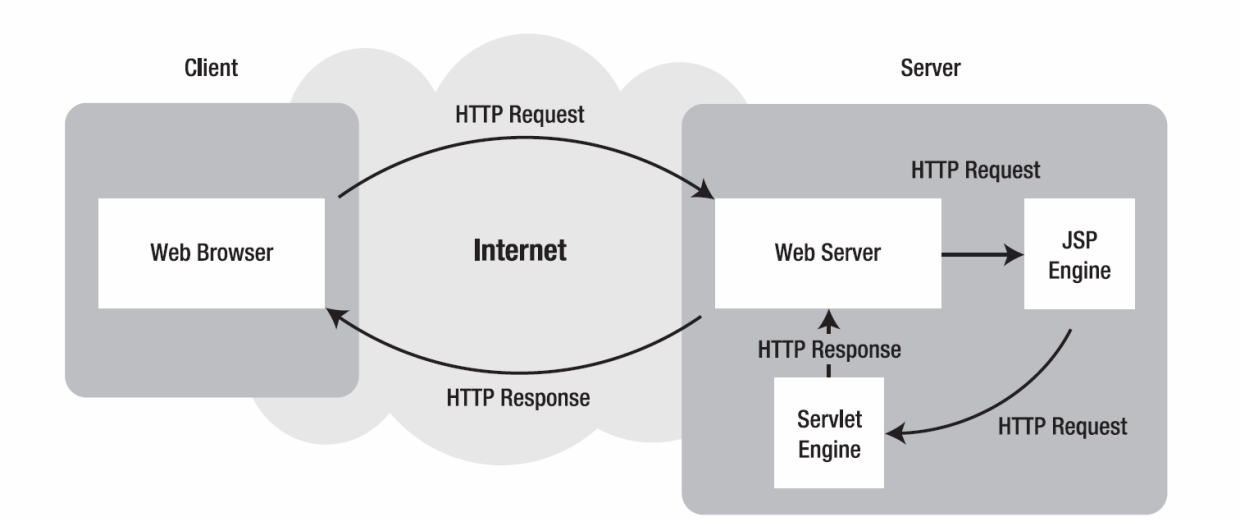
Slika 6: Struktura veb aplikacije

Java serverske stranice

Java veb tehnologije obuhvataju dva osnovna načina pravljenja dinamičkih sadržaja veb stranica: servlete i *Java* serverske stranice (engl. *Java Server Pages – JSP*). Pri implementaciji podsistema za vođenje beleški, kao i pri impementaciji sistema StudInfo, korišćena je tehnologija *JSP*.

Java Server Pages (*JSP*) je tehnologija koja omogućava dinamičko generisanje veb stranica, čiji sadržaj zavisi od zahteva koji je poslat veb serveru. Kompanija *Sun Microsystems* je uvela ovu tehnologiju 1999. godine kako bi omogućila efikasan i lako prenosiv mehanizam razvoja veb stranica sa dinamičkim sadržajem. Tehnologija *JSP* predstavlja nadgradnju tehnologije *Java Servleta*, koja omogućava brži razvoj, održavanje i veću robusnost veb aplikacija. Paket *servlet* definiše klase u programskom jeziku *Java* pomoću kojih se predstavljaju *HTTP* zahtevi koji se šalju serveru preko veb pretraživača i *HTTP* odgovori koji se od servera šalju pretraživaču. *Servlet* predstavlja objekat koji se nalazi na serveru i prima *HTTP* zahteve poslate preko Interneta, pristupa resursima (npr. bazi podataka), priprema *HTTP* odgovor i šalje *HTTP* odgovor.

Kada se preko pretraživača zahteva da se pristupi nekoj veb strani, izvode se sledeći koraci (predstavljeni na slici 7):



Slika 7: Prikazivanje JSP stranice

1. Veb pretraživač šalje *HTTP* zahtev (eng. *HTTP Request*) veb serveru.
2. Veb server sadrži potrebne dodatke za identifikovanje i rukovanje *Java* servletima. Veb server prepoznaje da se pristigli *HTTP* zahtev odnosi na *JSP* stranicu i prosleđuje ga *JSP* mašini.
3. *JSP* mašina čita *JSP* stranicu sa diska i konvertuje je u *Java* servlet. Taj servlet je automatski generisan i teško čitljiv, ali se po načinu rada ne razlikuje od servleta koji je razvijen neposredno u programskom jeziku *Java.*
4. *JSP* mašina kompajlira servlet u izvršnu klasu i prosleđuje zahtev servlet mašini. *JSP* mašina samo prevodi *JSP* stranicu u *Java* kod i po potrebi ponovo kompajlira servlet samo ako je *JSP* stranica promenjena od poslednjeg zahteva.
5. Deo veb servera koji se naziva servlet mašina izvršava prevedenu *Servlet* klasu, i za vreme izvršavanja servleta generiše izlaz u odgovarajućem formatu (najčešće *HTML*, ali može biti i drugi tekstualni ili bilo koji binarni format, na primer *JPEG*, *MP3* i drugo), koji servlet mašina prosleđuje veb serveru u okviru *HTTP* odgovora (eng. *HTTP Response)*.
6. Veb server prosleđuje *HTTP* odgovor pretraživaču koji je poslao zahtev.
7. Veb pretraživač obrađuje dinamički generisanu *HTML* stranu u okviru *HTTP* odgovora na isti način kao da je u pitanju statička strana.

Isti servlet može da pravi različite izlaze, zavisno od parametara *HTTP* zahteva i drugih faktora.

JSP stranica je tekstualni dokument koji se sastoji od dva tipa elemenata: statičkih elemenata, koji mogu biti u bilo kom tekstualnom formatu (*HTML, SVG, WML* ili *XML*) i specifičnih *JSP* elemenata, koji predstavljaju uputstva za pravljenje dinamičkih delova sadržaja. Preporučena ekstenzija *JSP* stranice je *jsp*. U jednu stranicu mogu da se umetnu druge datoteke, koje su *JSP* stranice ili delovi *JSP* stranice. Preporučena ekstenzija za datoteke koje sadrže samo deo *JSP* stranice je *jspf*. *JSP* elementi u *JSP* stranici mogu biti navedeni primenom jedne od dve

sintakse: standardne (*HTML*) sintakse i *XML* sintakse. Svaka datoteka može da koristi samo jednu sintaksu.

Postoje tri tipa *JSP* elemenata:

- direktive – obezbeđuju informacije potrebne za prevođenje stranice. Kako nemaju veze sa pojedinačnim zahtevom, ne učestvuju u pravljenju delova *HTML* stranice koja se vraća kao odgovor.
- akcije – definišu akcije koje se izvode kada se zahteva stranica. Mogu da koriste, menjaju i/ili prave objekte, i mogu da utiču na koji će se način prikazati podaci.
- skriptovi – delovi koda napisani na programskom jeziku *Java* koji rukuju *Java* objektima. Izvršavaju se kada se stranica zahteva, i mogu da urade neki posao, kao što je pristupanje resursima, ili izračunaju i dodaju neke elemente stranici koja se šalje i prikazuje korisniku. *JSP* skript elementi omogućavaju da se koriste izrazi programskog jezika *Java* u *JSP* stranicama. Skript elementi se koriste za pravljenje i pristupanje objektima ili definisanje metoda klase koja odgovara toj stranici.

Direktive

Direktive se izvršavaju kada se *JSP* stranica prevodi u *Java* kod. Njihova sintaksa je :

```
<%@ directive-name attr1="value1" [attr2="value2"...] %>
```

gde *directive-name* može biti:

- *page* – koristi se za definisanje svojstava koji se odnose na *JSP* stranicu i određuju se putem atributa. Može da se navede više direktiva *page* u jednoj *JSP* stranici, i sve će biti primenjene. Redosled i mesto pojavljivanja na stranici je nevažno. Najčešće strana počinje sledećom direktivom:

  ```
  <%@page language="java" contentType="text/html"%>
  ```

  a zatim se najčešće navodi koje su *Java* biblioteke ili klase potrebne, npr.:

  ```
  <%@page import="java.io.*, java.util.*"%>
  ```

  Neki od atributa koje podržava direktiva *page* su:

  - *contentType* – definiše *MIME* tip i kodnu stranu koju koristi *JSP* stranica;
  - *errorPage* – definiše *JSP* stranicu kojoj se šalju izuzeci ukoliko nastanu;
  - *import* – definiše pakete programskog jezika *Java* koje treba uključiti u *JSP* stranicu;

- *isErrorPage* – definiše da li stranica služi za obradu grešaka;
- *language* – definiše koji programski jezik se koristi u *JSP* stranici.

- *include* – umeće sadržaj druge datoteke u *JSP* stranicu. Aplikativni server *Tomcat* umeće sadržaje pre prevođenja strane u *Java* kod i servlet klasu. Na primer, ako kod uključuje sadržaj datoteke koja se zove *umetak* sa ekstenzijom *ext*, koja je u istom direktorijumu kao i stranica u koju se umeće onda direktiva izgleda:

```
<%@include file="umetak.ext"%>
```

- *taglib* – pored postojećih *JSP* etiketa mogu se definisati nove etikete, tako što se dodaju nove biblioteke etiketa i definiše prefiks za svaku od njih, kako bi se razlikovale nove etikete od postojećih. Na primer, direktivom:

```
<%@taglib uri="http://mysite.com/mytags" prefix="my" %>
```

se omogućava da se koriste etikete oblika :

```
<my:oneOfMyTags> ... </my:oneOfMyTags>
```

Akcije

Svrha akcija je da se odredi akcija koja će se izvršiti kada se zahteva stranica. Uobičajena sintaksa akcija je:

```
<jsp:action-name action-attribute-list/>
```

Neke od standardnih akcija su:

- *element* za dinamičko generisanje *XML* elementa;
- *forward* za prosleđivanje *HTTP* zahteva drugoj *JSP* ili *HTML* stranici ili servletu;
- *include* za uključivanje statičke ili dinamičke datoteke u *JSP* stranicu.

Skript elementi

Postoje tri tipa skript elemenata:

- deklaracije – koriste se za deklaraciju promenjivih i metoda koji se koriste u *JSP* stranici. Sintaksa deklaracije je:

```
<%! deklaracije %>
```

- izrazi – koriste se za dodavanje vrednosti dobijene izračunavanjem izraza na programskom jeziku *Java* u *HTML* kod. Rezultat izraza se

prevodi u objekat klase *String* koji se zatim prikazuje. Sintaksa je oblika:

```
<%= izraz %>
```

Na primer:

```
...

Server date and time: <%= new Date() %>

...
```

- skriptleti predstavljaju ispravan programski kod na programskom jeziku *Java* koji se ugrađuje u *JSP* stranicu. Sintaksa je oblika:

```
<% programski kod na Javi %>
```

U okviru skriptleta se mogu definisati nove promenjive, koristiti promenjive i metode definisane u deklaracijama ili umetnutim stranicama, uključene klase i biblioteke kao i implicitni objekti.

U skriptovima se mogu koristiti i implicitni objekti koje pravi aplikativni server (npr. *Tomcat*) koji sadrže informacije o određenom zahtevu, stranici, sesiji ili aplikaciji. Neki od najčešće upotrebljivanih implicitnih objekata su:

- *application* – objekat aplikacije, koji omogućava pristup resursima koji se dele unutar veb aplikacije;
- *config* – objekat konfiguracije, koji aplikativni server koristi za prosleđivanje informacija servletima;
- *exception* – objekat iuzetka, dostupan je samo u stranicama namenjenim za obradu greške;
- *pageContext* – objekat okruženja stranice, koji omogućava pristup svim objektima i atributima *JSP* stranice;
- *out* – objekat izlaznog toka, u koji se ispisuje kod stranice koji se izračunava, pri upotrebi izraza se rezultat izraza ispisuje u tok *out;*
- *request* – objekat kojim se predstavlja *HTTP* zahtev koji je poslat aplikativnom serveru od strane klijenta i na osnovu koga je započelo izračunavanje stranice;
- *response* – objekat kojim se predstavlja odgovor na *HTTP* zahtev. Objekat pored sadržaja (koji se ispisuje u tok *out*) sadrži i druge informacije koje se šalju klijentu (npr, kolačići, sadržaja *MIME* tipa i drugo);
- *session* – objekat korisničke sesije kojoj pripada *HTTP* zahtev koji se obrađuje. Objekat sadrži informacije koje se odnose na klijenta koji šalje *HTTP* zahtev.

Postoji i nekoliko objekata koji omogućavaju lakši pristup nekim elementima *HTTP* zahteva:

- *param* - preslikava ime parametra zahteva u jedan objekat klase *String*;

- *paramValues* - preslikava ime parametra zahteva u niz objekata klase *String*;
- *header* - preslikava ime zaglavlja zahteva u jedan objekat klase *String*;
- *headerValues* - preslikava ime zaglavlja zahteva u niz objekata klase *String*.

Tomcat

Informacioni sistem StudInfo u implementaciji koristi aplikativni server *Apache Tomcat*.

Tomcat je nastao kao projekat kompanije *Sun Microsystems*, kako bi se napravio primer implementacije specifikacije *Java Servlet*-a i *JSP*-a. Softverski arhitekta *James Duncan Davidson*, koji je radio na tom projektu, je 1999. godine pomogao da projekat postane otvorenog koda donacijom fondaciji *Apache Software Foundation*. Pošto je *James Duncan Davidson* od početka želeo da projekat bude otvorenog koda, ime mu je dao po životinji (tomcat na srpskom znači mačak) jer većina projekata otvorenog koda, koja je opisana knjigama izdavača *O'Reilly,* na naslovnoj strani ima neku životinju. Dao mu je ime *Tomcat* jer mačak predstavlja životinju koja može da se brine o sebi i brani od neprijatelja. Njegova želja se ostvarila.

Tomcat je servlet kontejner za *Java* servlete i *JSP* stranice. Servlet kontejner je termin koji u okviru tehnologija *JavaServlet* i *JSP* odgovara pojmu aplikativnog servera. Servlet kontejner predstavlja program koji omogućava učitavanje, inicijalizaciju i izvršavanje servleta, odnosno obezbeđuje *Java* virtualnu mašinu i potrebne elemente za potpuno izvršno okruženje *Java* programskog koda. Takođe obezbeđuje i potreban softver za veb server kako bi okruženje bilo dostupno na Internetu.

Kako je implementacija *Tomcat*-a zasnovana na *Java Servlet* i *JSP* specifikaciji, on predstavlja standard prema kome se mere proizvodi proizvođača kontejnera za servlete i *JSP* stranice. Kod koji radi pod *Tomcat*-om, morao bi da radi i pod ostalim kontejnerima koji odgovaraju standardu.

Hibernate

U implementaciji informacionog sistema StudInfo koristi se relacioni sistem za upravljanje bazamam podataka *IBM DB2*. Pri upotrebi relacione baze podataka iz objektno orijentisane aplikacije javlja se problem kako objekte klasa programskog jezika *Java* preslikati u redove tabela relacione baze podataka, ili kako klase programskog jezika *Java* preslikati u relacionu shemu i obrnuto. Kako bi se rešio taj problem koristi se tehnika objektno-relacionog preslikavanja između entiteta u relacionoj bazi i objekata u objektno orijentisanim programskim jezicima.

Hibernate je alat otvorenog koda koji služi za objektno-relaciono preslikavanje u programskom jeziku *Java*. Omogućava trajno čuvanje i pretraživanje objekata kao i čuvanje definisanih asocijacija, nasleđivanja, polimorfizama, kompozicija i kolekcija.

Neke od glavnih osobina alata *Hibernate* su:

- objektna i relaciona preslikavanja;
- objektno-orijentisan upitni jezik;
- automatsko generisanje primarnih ključeva;
- visoke performanse.

Hibernate je napisan na programskom jeziku *Java* i ima dva tipa konfiguracionih datoteka. Prva od njih je datoteka *hibernate.cfg.xml*. To je datoteka u formatu *XML*, koja sadrži operativna svojstva, kao što su informacije potrebne za povezivanje sa bazom podataka, lozinka, dijalekt baze podataka i lokacija datoteka za preslikavanje. Drugi tip konfiguracionih datoteka služi za opisivanje konkretnih preslikavanja (sa ekstenzijom **.hbm*) i sadrži uputstva kako da se izvrši preslikavanje između određene klase programskog jezika *Java* i jedne ili više tabela u bazi podataka. Obično jedna aplikacija ima jednu datoteku *hibernate.cfg.xml* i nekoliko *.hbm* datoteka.

1.3 Primenjeni alati

Pri projektovanju i implementiranju podsistema za beleške korišćeni su sledeći alati:

- *Visual Paradigm Business Process Visual ARCHITECT 4.0 Simulacian* za izradu dijagrama toka podataka [1]
- *Visual Paradigm for UML 8.0 Community Edition* za izradu dijagrama slučajeva upotrebe [2]
- *Visual Paradigm Database Visual ARCHITECT* 5.2 za izradu dijagrama entiteta i odnosa [3]
- Razvojno okruženje *Eclipse* [4] za pisanje programskog koda (JSP stranica i klasa u programskom jeziku Java).

2 Analiza

2.1 Definisanje zahteva

Svaki nastavnik povremeno ima potrebu da zapiše belešku ili da napravi podsetnik o kursu u čijoj nastavi učestvuje ili o nekom studentu. Beleške i podsetnike nastavnici obično vode na papiru ili u datoteci koju čuvaju na računaru. Takav način vođenja beleški i podsetnika je nepraktičan jer ih je lako izgubiti ili zaboraviti da postoje. Kako nastavnici neke poslove u vezi kurseva i studenata obavljaju na različitim lokacijama (Matematički fakultet ima dva međusobno udaljena objekta, a nastavnici mogu sređivati podatke o kursevima i studentima i od svoje kuće), postoji i problem dostupnosti i ažurnosti tako vođenih beleški. Zbog toga je jedno od mogućih unapređenja sistema StudInfo, kojim bi se dodatno olakšao proces vođenja podataka o nastavnim aktivnostima i studentima, upravo dodavanje mogućnosti vođenja beleški i podsetnika uz kurseve i studente. U oblikovanju polazne specifikacije zahteva učestvovali su mentor i članovi razvojnog tima sistema StudInfo.

Prvobitna specifikacija zahteva je navedena u okviru sistema za praćenje projekta:

Omogućiti nastavnicima da vode beleške uz određeni kurs.
** Svaka beleška je jedan uređeni tekstualni podatak.*
** Beleške se mogu dodavati, brisati i menjati.*
** Može se menjati redosled beleški (da li?).*
** Beleške se vode uz kurs.*
** Nastavnik ima mogućnost da vidi beleške koje je vodio uz prethodne kurseve za isti predmet.*

Omogućiti nastavnicima da vode beleške uz određene studente.
** Svaka beleška je jedan uređeni tekstualni podatak.*
** Beleške se mogu dodavati, brisati i menjati.*
** Može se menjati redosled beleški (da li?).*
** Beleške se vode uz podatke o studentu, nezavisno od kursa.*

Verovatno je dobro da se koncept beleške može vezati za jednog ili više studenata ili jedan ili više kurseva. U tom kontekstu, beleška je samostalan entitet, a za šta će se vezivati zavisi od toga odakle se pokreće njeno pravljenje. S tim da se i eksplicitno može promeniti njeno vezivanje (dodavanje kurseva i/ili studenata).

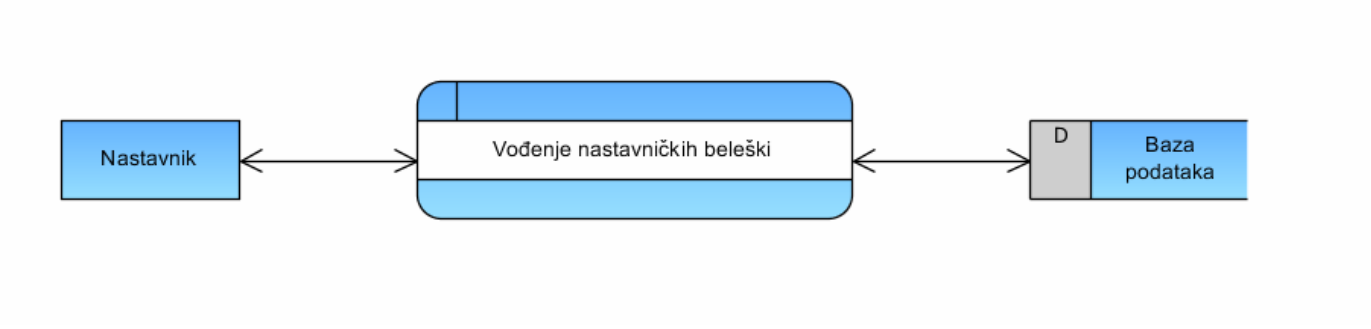
Beleška može da bude označena kao podsetnik. U tom slučaju se prikazuje svaki put pošto se nastavnik prijavi na sistem. Kratak pregled beleški se vidi na stranicama posvećenim grupama i kursevima, slično kao što je sada sa obaveštenjima. Beleške o studentima se vide samo pri otvaranju podataka o pojedinačnim studentima.

Omogućiti traženje beleški po kursevima, datumu, imenu ili indeksu studenta i tekstu beleške (samo primena operatora LIKE).

Da bi se zahtevi potpunije odredili, vođeno je više razgovora sa članovima razvojnog tima sistema StudInfo i potencijalnim korisnicima podsistema. Jedna od usvojenih dopuna je mogućnost označavanja beleške kao lične ili službene. Ličnu belešku može da vidi samo nastavnik koji ju je postavio, a službenu mogu da vide svi nastavnici koji učestvuju u održavanju nastave na kursevima ili koji predaju studentima na koje se beleška odnosi. Time se obezbeđuje bolja komunikacija među nastavnicima koji učestvuju u nastavi na istom kursu ili predaju istim studentima. Redosled beleški prilikom prikaza se određuje prema datumu njihovog postavljanja ili poslednje izmene. Zbog toga je potrebno obezbediti da prlikom ažuriranja beleške postoji mogućnost ažuriranja i datuma beleške. Prilikom pretraživanja beleške navodi se period u kom je postavljena tražena beleška, a ne jedan određen datum, da bi se omogućila lakša pretraga. Ponuđeni periodi mogu da budu: poslednjih 30 dana, tekuća školska godina i poslednjih pet godina.

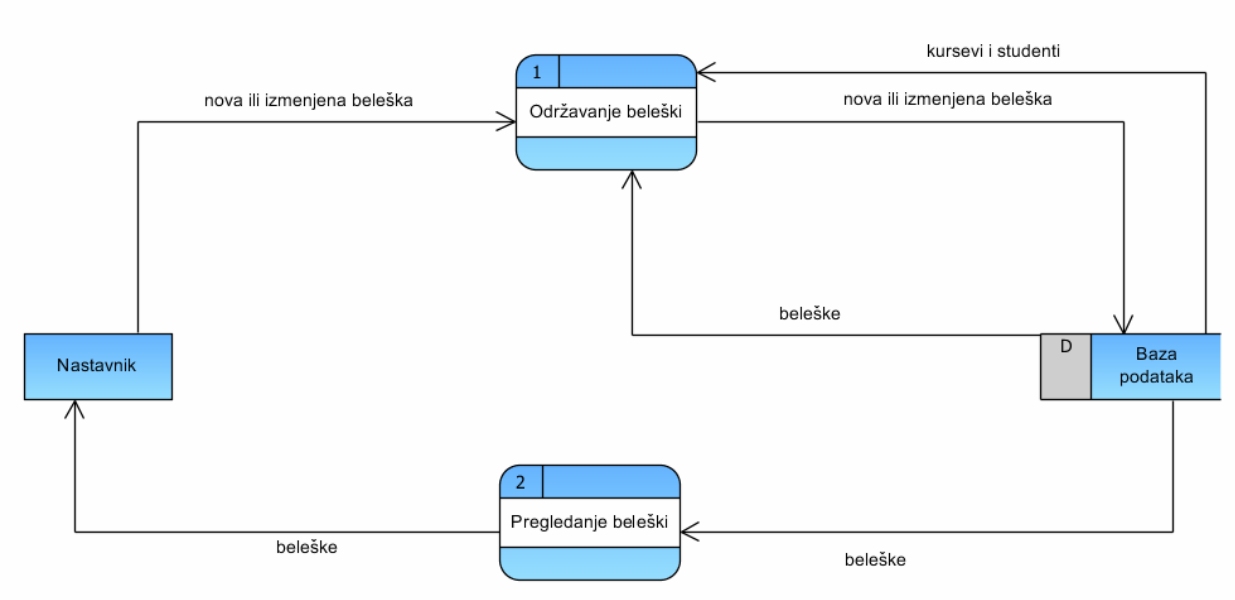
2.2 Dijagrami toka podataka

Na dijagramu konteksta podsistema za nastavničke beleške, svi procesi u podsistemu su predstavljeni jednim procesom (*Vođenje nastavničkih beleški*). Na dijagramu je prikazano sa kojim spoljašnjim entitetima (*Nastavnik*) i skladišta podataka (*Baza podataka*) interaguje podsistem.



Slika 8: Dijagram konteksta podsistema za vođenje nastavničkih beleški

Na dijagramu prvog nivoa proces iz dijagrama konteksta je podeljen na glavne procese koji se obavljaju u podsistemu za nastavničke beleške (*Održavanje beleški* uz kurseve i studente i *Pregledanje beleški*). Prikazani su i tokovi podataka između glavnih procesa, spoljašnjeg entiteta (*Nastavnik*) i skladišta podataka (*Baze podataka*).

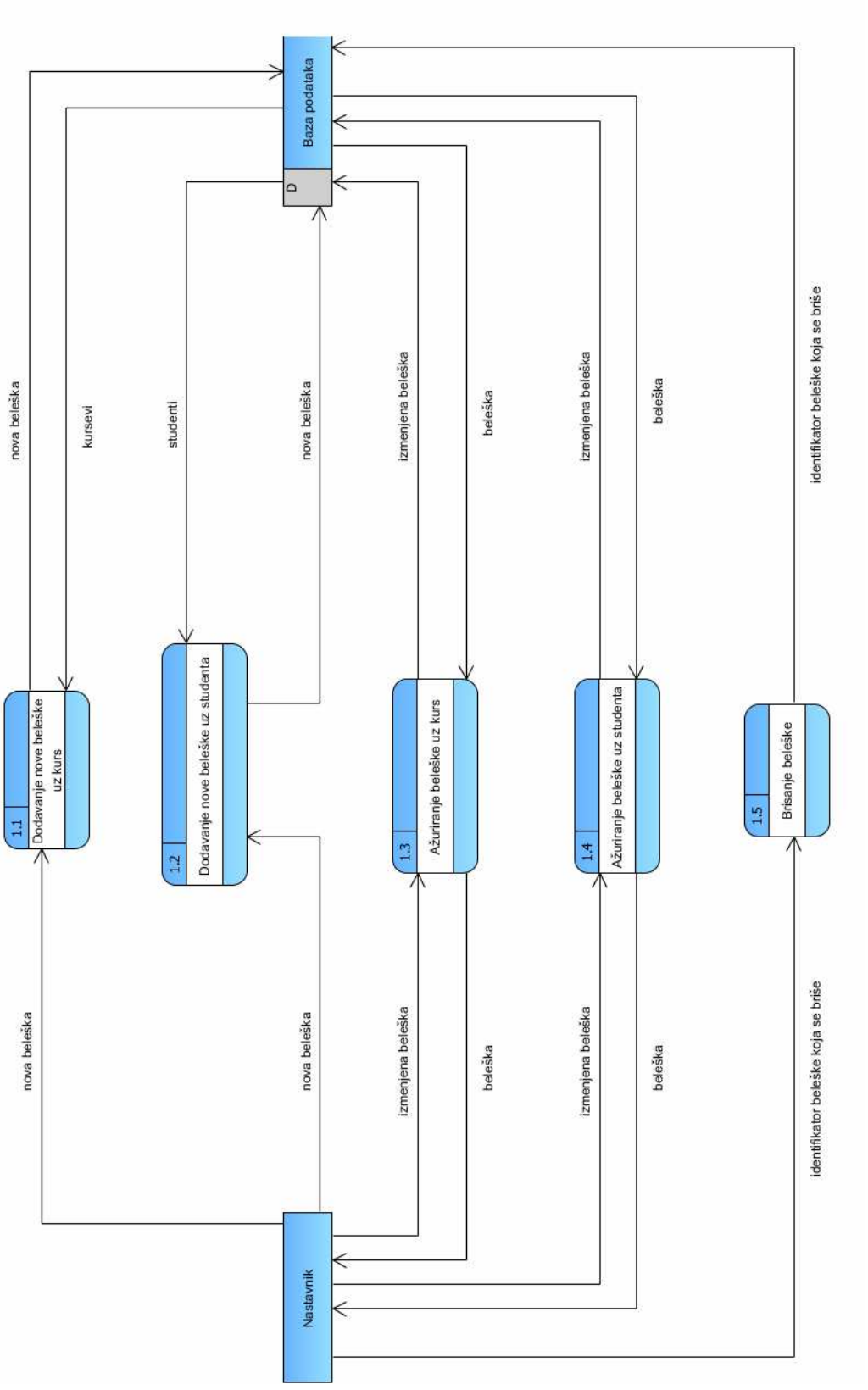


Slika 9: Dijagram prvog nivoa podsistema za nastavničke beleške

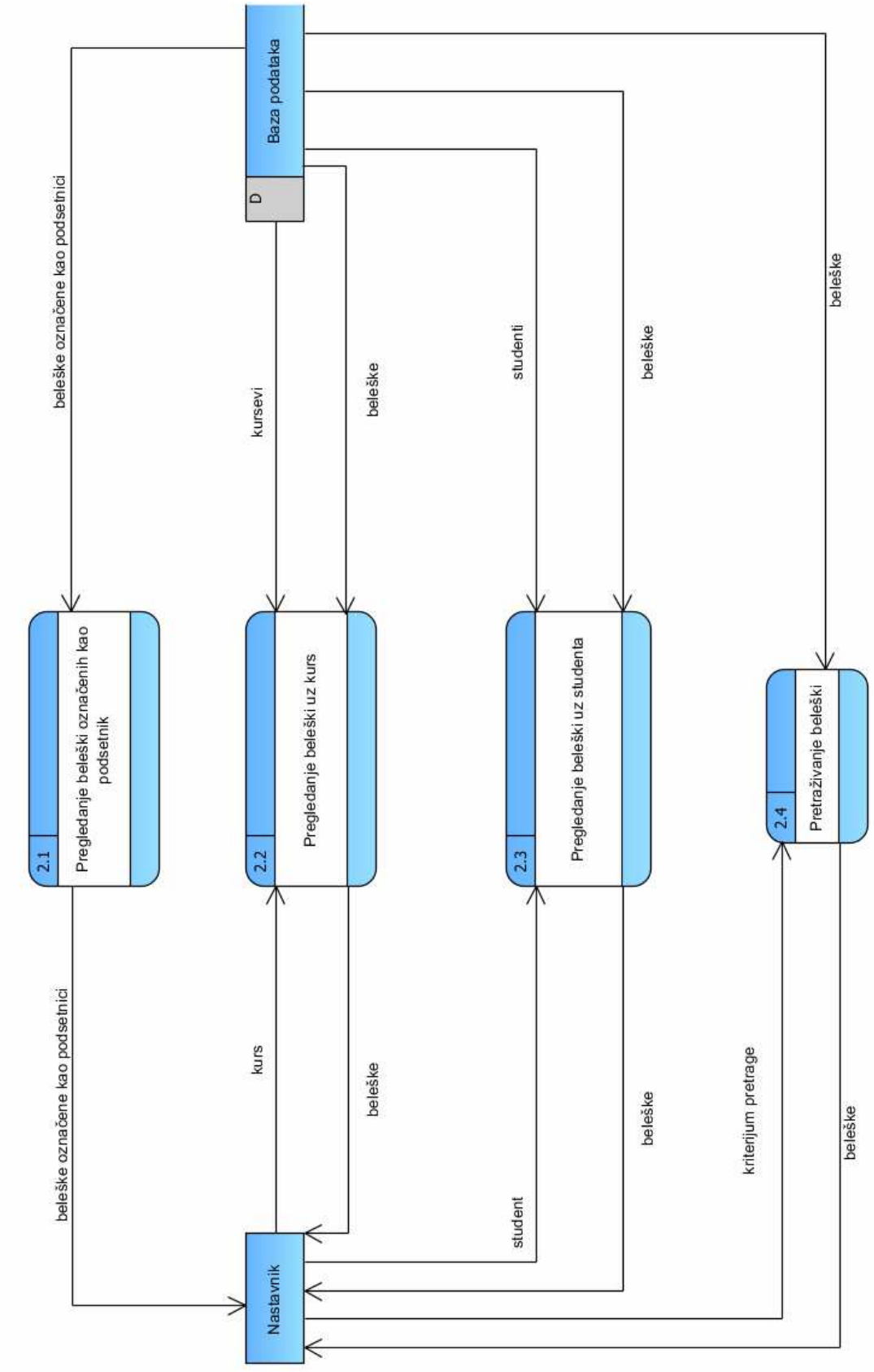
Na dijagramima drugog nivoa detaljnije su prikazani procesi od kojih se sastoje glavni procesi na dijagramu prvog nivoa, kao i tokovi podataka između procesa, spoljašnjeg entiteta (*Nastavnik*) i skladišta podataka (*Baza podataka*).

Održavanje beleški se sastoji od dodavanja nove beleške uz kurs, dodavanja nove beleške uz studenta, ažuriranja postojeće beleške uz kurs, ažuriranja postojeće beleške uz studenta, kao i brisanja postojeće beleške.

Pregledanje beleški je moguće ostvariti pregledanjem beleški označenih kao podsetnik, pregledanjem beleški uz kurs, pregledanjem beleški uz studenta i pretraživanjem svih beleški po različitim kriterijumima.

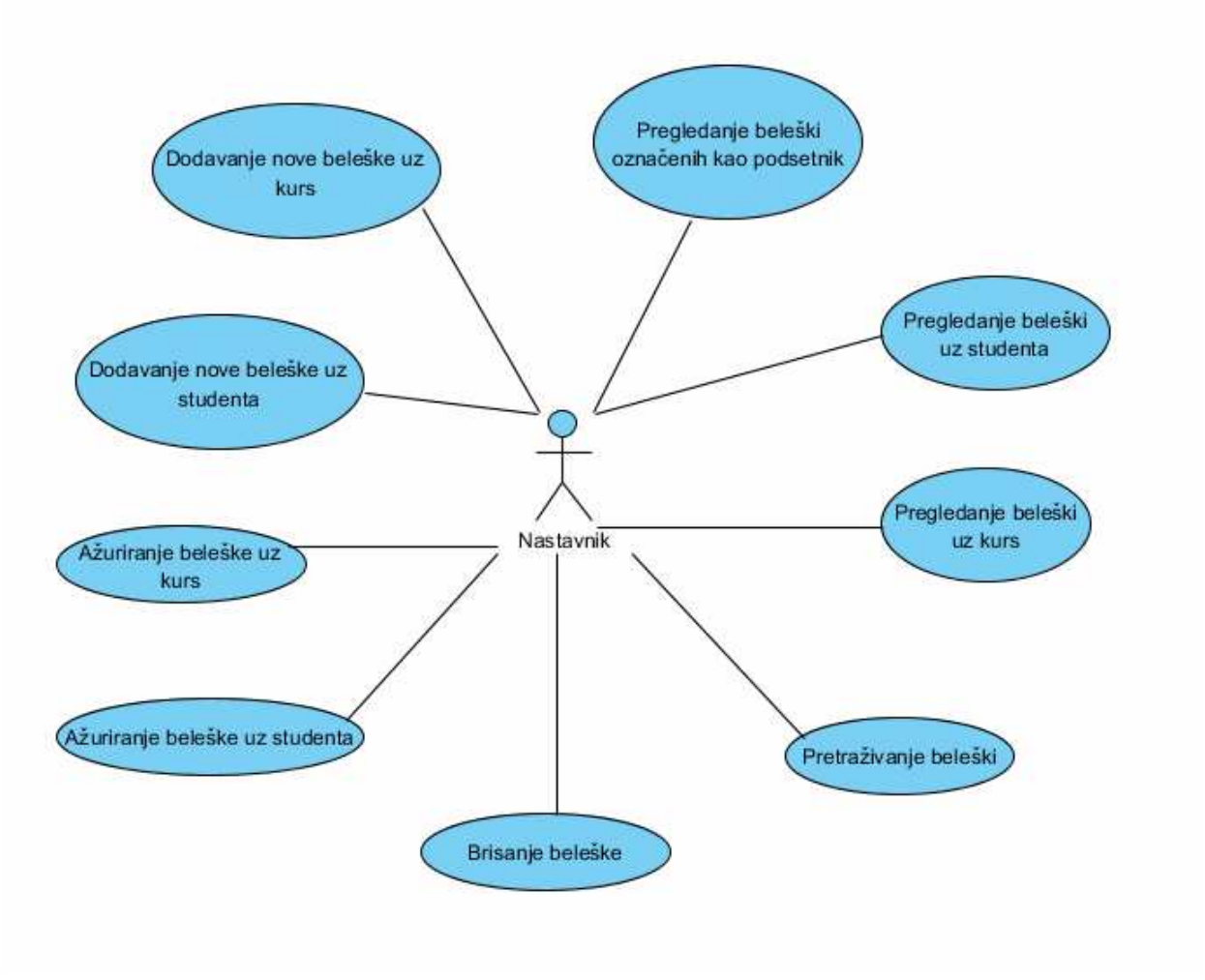


Slika 10: Dijagram drugog nivoa - održavanje beleški uz kurseve i studente



Slika 11: Dijagram drugog nivoa - pregledanje beleški

2.3 Slučajevi upotrebe



Slika 12: Dijagram slučajeva upotrebe podsistema za nastavničke beleške

U okviru podsistema za nastavničke beleške prepoznato je 11 slučajeva upotrebe:

1. Dodavanje nove beleške uz kurs
2. Dodavanje nove beleške us studenta
3. Pregledanje beleški uz kurs
4. Pregledanje beleški uz studenta
5. Ažuriranje beleške uz kurs
6. Ažuriranje beleške uz studenta
7. Brisanje postojeće beleške
8. Pregledanje beleški označenih kao podsetnika
9. Pretraživanje beleški.

U nastavku će biti opisan svaki od navedenih slučajeva upotrebe.

2.3.1 Dodavanje nove beleške uz kurs

1. **Kratak opis**: Nastavnik bira kurs uz koji želi da postavi novu belešku i popunjava podatke o novoj belešci i određuje uz koje kurseve će se

postaviti. Belešku može da označi kao podsetnik i takve beleške se prikazuju svaki put kada se prijavi na sistem.

2. **Učesnik**: Nastavnik koji ima pravo da postavi belešku uz izabrani kurs.

3. **Tok događaja**

   3.1. **Osnovni tok**

   - Nastavnik bira opciju dodavanja nove beleške uz izabrani kurs.
   - Nastanik unosi tekst beleške.
   - Nastavnik definiše da li beleška treba da bude označena kao podsetnik.
   - Nastavnik definiše belešku kao ličnu ili službenu, odnosno da li može samo on da vidi tu belešku ili mogu da vide svi nastavnici koji učestvuju u održavanju nastave iz kursa uz koji se postavlja beleška.
   - Nastavnik ima mogućnost da izabere još kurseva koje drži u tekućoj školskoj godini, tako da se nova beleška odnosi i na njih. Nastavnik može i da isključi neki od kurseva koje je već izabrao, kako se beleška ne bi odnosila na njih.
   - Nastavnik potvrđuje dodavanje beleške.
   - Sistem prikazuje izveštaj o izvršenim promenama.

   3.2. **Alternativni tokovi**

   - **Nedostatak podataka**: Ukoliko nastavnik pokuša da sačuva belešku, a nije uneo tekst beleške i izabrao bar jedan kurs, sistem izdaje odgovarajuću poruku i zahteva da se ti uslovi ispune pre nego što se beleška sačuva.

   - **Poništavanje izmena**: Ukoliko nastavnik izabere opciju poništavanja izmena, svi uneti podaci o belešci biće poništeni.

   - **Prekid unosa**: Ukoliko nastavnik izabere opciju da otvori neku drugu stranicu automatski se obustavlja dodavanje beleške i uneti podaci neće biti zapamćeni.

4. **Preduslovi**: Nastavnik može da postavlja beleške samo uz kurseve koje drži u tekućoj školskoj godini. Na otvorenoj stranici postoji opcija dodavanja nove beleške uz neki kurs.

5. **Postuslovi**: Nova beleška je upisana u bazu podataka. Otvorena je stranica za prikazivanje dodate beleške.

2.3.2 Dodavanje nove beleške uz studenta

1. **Kratak opis**: Nastavnik bira studenta uz kojeg želi da postavi novu belešku, popunjava podatke o novoj belešci i određuje skup studenta uz koje će se beleška postaviti. Belešku može da označi kao podsetnik i takve beleške se prikazuju svaki put kada se prijavi na sistem.

2. **Učesnik**: Nastavnik koji ima pravo da postavi belešku uz izabranog studenta.

3. **Tok događaja**

   3.1. **Osnovni tok**

   - Nastavnik bira opciju dodavanja nove beleške uz izabranog studenta.
   - Nastanik unosi tekst beleške.
   - Nastavnik definiše da li beleška treba da bude označena kao podsetnik.
   - Nastavnik definiše belešku kao ličnu ili službenu, odnosno da li može samo on da vidi tu belešku ili mogu da vide svi nastavnici koji predaju studentu uz kojeg se postavlja beleška.
   - Nastavnik ima mogućnost da izabere još studenata kojima predaje u tekućoj školskoj godini, tako da se nova beleška odnosi i na njih. Nastavnik može i da isključi nekog od studenata koje je već izabrao, kako se beleška ne bi odnosila na njih.
   - Nastavnik potvrđuje dodavanje beleške.
   - Sistem prikazuje izveštaj o izvršenim promenama.

   3.2. **Alternativni tokovi**

   - **Nedostatak podataka**: Ukoliko nastavnik pokuša da sačuva belešku, a nije uneo tekst beleške i izabrao bar jedog studenta, sistem izdaje odgovarajuću poruku i zahteva da se ti uslovi ispune pre nego što se beleška sačuva.
   - **Poništavanje izmena**: Ukoliko nastavnik izabere opciju poništavanja izmena, svi uneti podaci o bečešci biće poništeni.
   - **Prekid unosa**: Ukoliko nastavnik izabere opciju da otvori neku drugu stranicu automatski se obustavlja dodavanje beleške i uneti podaci neće biti zapamćeni.

4. **Preduslovi**: Nastavnik može da postavlja beleške samo uz studente kojima predaje u tekućoj školskoj godini. Na otvorenoj stranici postoji opcija dodavanja nove beleške uz nekog studenta.

5. **Postuslovi**: Nova beleška je upisana u bazu podataka. Otvorena je stranica za prikazivanje dodate beleške.

2.3.3 Pregledanje beleški uz kurs

1. **Kratak opis**: Nastavnik bira kurs za koji želi da vidi beleške koje je postavio uz njega ili su postavljene kao službene. Prikazuju mu se beleške.

2. **Učesnik**: Nastavnik koji ima pravo da vidi bar jednu belešku postavljenu uz izabrani kurs.

3. **Tok događaja**

   3.1. **Osnovni tok**

   Nastavnik bira opciju pregledanja beleški uz kurs.
   Za svaku belešku se prikazuje:
   - datum postavljanja;
   - ime i prezime nastavnika koji je postavio;
   - tekst beleške.

   Za svaku belešku se prikazuje lista svih kurseva uz koje je beleška postavljena.
   Za svaku belešku koju je postavio prijavljeni nastavnik daje se mogućnost izmene i brisanja beleške.

   3.2. **Alternativni tokovi**

   Nema beleški: Ako uz izabrani kurs nema beleški koje nastavnik ima pravo da vidi, prikazuje se odgovarajuća poruka.

4. **Preduslovi**: Nastavnik može da pregleda beleške samo uz kurseve u čijoj nastavi učestvuje. Otvorena je stranica koja se odnosi na neki kurs i na njoj postoji opcija pregledanja postojećih beleški uz kurs.

5. **Postuslovi**: Otvorena je stranica za prikazivanje beleški uz izabrani kurs.

2.3.4 Pregledanje beleški uz studenta

1. **Kratak opis**: Nastavnik bira studenta uz kojeg želi da vidi beleške koje je postavio uz njega ili su postavljene kao službene. Prikazuju se beleške uz izabranog studenta.
2. **Učesnik**: Nastavnik koji ima pravo da vidi bar jednu belešku postavljenu uz izabranog studenta.

3. **Tok događaja**

   3.1. **Osnovni tok**

   Nastavnik bira opciju pregledanja beleški uz studenta.
   Za svaku belešku se prikazuje:
   - datum postavljanja;
   - ime i prezime nastavnika koji je postavio;
   - tekst beleške.

   Za svaku belešku se prikazuje lista svih studenata uz koje je beleška postavljena.
   Za svaku belešku koju je postavio prijavljeni nastavnik daje se mogućnost izmene i brisanja beleške.

3.2. **Alternativni tokovi**

Nema beleški: Ako uz izabranog studenta nema beleški koje nastavnik ima pravo da vidi, prikazuje se odgovarajuća poruka.

4. **Preduslovi**: Nastavnik može da pregleda beleške samo uz studenta kojem predaje. Otvorena je stranica koja se odnosi na nekog studenta i na njoj postoji opcija pregledanja postojećih beleški uz studenta.
5. **Postuslovi**: Otvorena je stranica za prikazivanje beleški uz izabranog studenta.

2.3.5 Ažuriranje beleške uz kurs

1. **Kratak opis**: Nastavnik ima mogućnost da menja belešku koju je postavio tokom tekuće školske godine uz kurs u čijoj nastavi učestvuje.
2. **Učesnik**: Nastavnik koji ima pravo da menja izabranu belešku.
3. **Tok događaja**

3.1. **Osnovni tok**

Bira se opcija menjaja postojeće beleške uz kurs.
Sistem prikazuje podatke izabrane beleške i daje mogućnost njihove promene. Prikazuju se:
- tekst beleške;
- da li je beleška označena kao podsetnik;
- tip beleške: lična ili službena;
- lista kurseva na koje se odnosi beleška;
- datum postavljanja beleške, moguće je izabrati da se datum ažuriranja beleške postavi kao datum postavljanja beleške.

Prikazuju se kursevi koji mogu da se dodaju kako bi se beleška odnosila i na njih.
Nastavnik potvrđuje izmene beleške.
Sistem prikazuje izveštaj o izvršenim promenama.

3.2. **Alternativni tokovi**

Nedostatak podataka: Ukoliko nastavnik pokuša da sačuva promene, a nije uneo tekst beleške i izabrao bar jedan kurs, sistem izdaje odgovarajuću poruku i zahteva da se ti uslovi ispune pre nego što se promene sačuvaju.

Poništavanje izmena: Ukoliko nastavnik izabere opciju poništavanja izmena, sve promene će biti poništeni.

Prekid unosa: Ukoliko nastavnik izabere opciju da otvori neku drugu stranicu automatski se obustavlja ažuriranje beleške i promenjeni podaci neće biti zapamćeni.

4. **Preduslovi**: Nastavnik može da menja samo beleške koje je on postavio u tekućoj školskoj godini. Otvorena je stranica koja prikazuje beleške uz kurseve i na njoj postoji opcija ažuriranja prikazane beleške.

5. **Postuslovi**: Promene su zabeležene u bazi podataka. Otvorena je stranica za prikazivanje ažurirane beleške.

2.3.6 Ažuriranje beleške uz studenta

1. **Kratak opis**: Nastavnik ima mogućnost da menja belešku koju je postavio uz studenta.
2. **Učesnik**: Nastavnik koji ima pravo da menja izabranu belešku.
3. **Tok događaja**

   3.1. **Osnovni tok**

   - Bira se opcija menjaja postojeće beleške uz studenta.
   - Sistem prikazuje podatke izabrane beleške i daje mogućnost njihove promene. Prikazuju se:
     - tekst beleške;
     - da li je beleška označena kao podsetnik;
     - tip beleške: lična ili službena;
     - lista studenata na koje se odnosi beleška;
     - datum postavljanja beleške, moguće je izabrati da se datum ažuriranja beleške postavi kao datum postavljanja beleške.
   - Prikazuju se studenti koji mogu da se dodaju kako bi se beleška koja se menja odnosila i na njih.
   - Nastavnik potvrđuje izmene beleške.
   - Sistem prikazuje izveštaj o izvršenim promenama.

   3.2. **Alternativni tokovi**

   - **Nedostatak podataka**: Ukoliko nastavnik pokuša da sačuva promene, a nije uneo tekst beleške i izabrao bar jednog studenta, sistem izdaje odgovarajuću poruku i zahteva da se ti uslovi ispune pre nego što se promene sačuvaju.

   - **Poništavanje izmena**: Ukoliko nastavnik izabere opciju poništavanja izmena, sve promene će biti poništeni.

   - **Prekid unosa**: Ukoliko nastavnik izabere opciju da otvori neku drugu stranicu automatski se obustavlja ažuriranje beleške i promenjeni podaci neće biti zapamćeni.

4. **Preduslovi**: Nastavnik može da menja samo beleške koje je on postavio. Otvorena je stranica koja prikazuje beleške uz studente i na njoj postoji opcija ažuriranja prikazane beleške.

5. **Postuslovi**: Promene su zabeležene u bazi podataka. Otvorena je stranica za prikazivanje ažurirane beleške.

2.3.7 Brisanje beleške

1. **Kratak opis**: Nastavnik ima mogućnost da briše belešku koju je postavio tokom tekuće školske godine.
2. **Učesnik**: Nastavnik koji ima pravo da briše izabranu belešku.
3. **Tok događaja**

   3.1. **Osnovni tok**

   - Nastavnik bira opciju brisanja postojeće beleške.
   - Sistem traži potvrdu brisanja izabrane beleške.
   - Nastavnik potvrđuje.
   - Sistem briše belešku.
   - Sistem prikazuje beleške bez prethodno obrisane beleške.

   3.2. **Alternativni tokovi**

   - **Odustajanje od brisanja**: Ukoliko se nastavnik predomisli i ne potvrdi brisanje, beleška ostaje nepromenjena.

4. **Preduslovi**: Nastavnik može da briše samo beleške koje je on postavio u tekućoj školskoj godini. Otvorena je stranica koja prikazuje beleške uz kurseve ili studente i na njoj postoji opcija brisanja beleški.
5. **Postuslovi**: Brisanje beleške je zabeleženo u bazi podataka. Otvorena je stranica za prikazivanje beleški.

2.3.8 Pregledanje beleški označenih kao podsetnik

1. **Kratak opis**: Nastavnik bira opciju da vidi beleške označene kao podsetnik koje je on postavio ili su postavljene kao službene, a on ima pravo da ih vidi. Prikazuju mu se beleške označene kao podsetnik.
2. **Učesnik**: Nastavnik koji ima pravo da vidi bar jednu belešku označenu kao podsetnik.
3. **Tok događaja**

   3.1. **Osnovni tok**

   - Nastavnik bira opciju pregledanja beleški označenih kao podsetnik.
   - Za svaku belešku označenu kao podsetnik se prikazuje:
     - datum postavljanja;
     - ime i prezime nastavnika koji je postavio;
     - tekst beleške.

Za svaku belešku označenu kao podsetnik se prikazuje lista svih kurseva ili studenata uz koje je beleška postavljena.
Za svaku belešku označenu kao podsetnik u toku tekuće školske godine daje se mogućnost izmene i brisanja.

3.2. **Alternativni tokovi**: Nema ih.

4. **Preduslovi**: Nastavnik može da pregleda samo beleške označene kao podsetnik koje je on postavio ili su postavljene kao službene, a on ima pravo da ih vidi. Otvorena je stranica na kojoj postoji opcija pregledanja postojećih beleški označenih kao podsetnik.

5. **Postuslovi**: Otvorena je stranica za prikazivanje beleški koje su označene kao podsetnici.

2.3.9 Pretraživanje beleški

1. **Kratak opis**: Nastavnik ima mogućnost da pretražuje beleške koje ima pravo da vidi po različitim kriterijumima.
2. **Učesnik**: Nastavnik koji je prijavljen na sistem.
3. **Tok događaja**

3.1. **Osnovni tok**

Nastavnik bira opciju pretraživanja beleški.
Prikazuju se podaci po kojima se definiše kriterijum pretraživanja. Kriterijum može da se definiše po:

- vrsti pretraživanja:
  - studenti i kursevi;
  - samo studenti;
  - samo kursevi.
- reči ili frazi koju sadrži tekst beleške;
- označenosti beleške, da li da se prikažu:
  - sve beleške;
  - samo podsetnici;
  - bez podsetnika.
- tipu beleške:
  - lične;
  - službene;
  - lične i službene.
- periodu u okviru kojeg je postavljena beleška:
  - u poslednjih 30 dana;
  - u tekućoj školskoj godini;
  - u poslednjih pet godina.
- kursevima, ako kursevi ulaze u pretragu. Daje se mogućnost izbora jednog kursa ili svih kurseva.
- studentima, ako studenti ulaze u pretragu. Daje se mogučnost pretrage po:
  - indeksu studenta;

- imenu studenta;
- prezimenu studenta.

Zadaje se kriterijum.

Prikazuju se beleške koje zadovoljavaju dati kriterijum.

3.2. **Alternativni tokovi**

Nema beleški koji zadovoljavaju kriterijum: Ukoliko ne postoje beleške koje zadovoljavaju kriterijum prikazuje se odgovarajuća poruka.

4. **Preduslovi**: Nema ih.

5. **Postuslovi**: Otvorena je stranica za prikazivanje beleški koje zadovoljavaju zadati kriterijum.

3 Projektovanje

3.1 Projektovanje baze podataka

Analizom slučajeva upotrebe napravljen je model entiteta i odnosa pri čemu su uočeni nezavisni entiteti:

1. ***Beleška*** – predstavlja jednu belešku koju je nastavnik postavio uz svoje kurseve ili studente.
2. ***Kurs*** – predstavlja jedno držanje nastave iz jednog predmeta, nastavnik može da beleške postavlja uz kurseve.
3. ***Dosije*** – predstavlja jednog studenta, nastavnik može da beleške postavlja uz studente.
4. ***Nastavnik*** – predstavlja osobu koja učestvuje u održavanju nastave (drži predavanja, vežbe ili praktikume) iz kurseva za koje se postavlja beleška ili studentima za koje se postavlja beleška.
5. ***Grupa*** – predstavlja jednu grupu kojoj nastavnik drži neki vid nastave iz nekog kursa (npr. grupa predavanja, vežbi ili praktikuma). Značajna je jer se preko grupa uspostavlja odnos nastavnik-student. Nastavnik predaje nekom studentu ako i samo ako nastavnik drži nastavu nekoj grupi kojoj taj student pripada.
6. ***Predmet*** – predstavlja jedan predmet koji se drži na kursevima. Jedan predmet može da se sluša na različitim kursevima u različitim školskim godinama.

Za navedene entitete uočeni su neophodni podaci koji će biti opisani uz svaki entitet.

Entitet ***Beleška*** opisuju sledeći podaci:

- tekst beleške;
- datum postavljanja ili menjanja beleške;
- informacija da li je beleška označena kao podsetnik ili ne;
- tip beleške, da li je beleška službena ili lična;
- nastavnik koji je posatavio belešku;
- na koje kurseve ili studente se beleška odnosi.

Entitet ***Kurs*** opisuju sledeći podaci:

- predmet koji se drži na kursu;
- školska godina u kojoj se drži kurs;
- fakultet na kome se sluša kurs;
- nastavnik koji je koordinator kursa;
- drugi podaci koji nisu značajni za vođenje beleški.

Entitet ***Dosije*** opisuju sledeći podaci:

- indeks studenta;
- fakultet na kome studira student;
- ime studenta;
- prezime studenta;
- drugi podaci koji nisu značajni za vođenje beleški.

Entitet ***Nastavnik*** opisuju sledeći podaci:

- fakultet na kome predaje nastavnik;
- ime nastavnika;
- prezime nastavnika;
- drugi podaci koji nisu značajni za vođenje beleški.

Entitet ***Grupa*** opisuju sledeći podaci:

- tip nastave koja se drži na grupi;
- kurs iz kojeg se drži navedeni tip nastave u grupi;
- nastavnik koji drži navedeni tip nastave u grupi;
- studenti grupe;
- drugi podaci koji nisu značajni za vođenje beleški.

Entitet ***Predmet*** opisuju sledeći podaci:

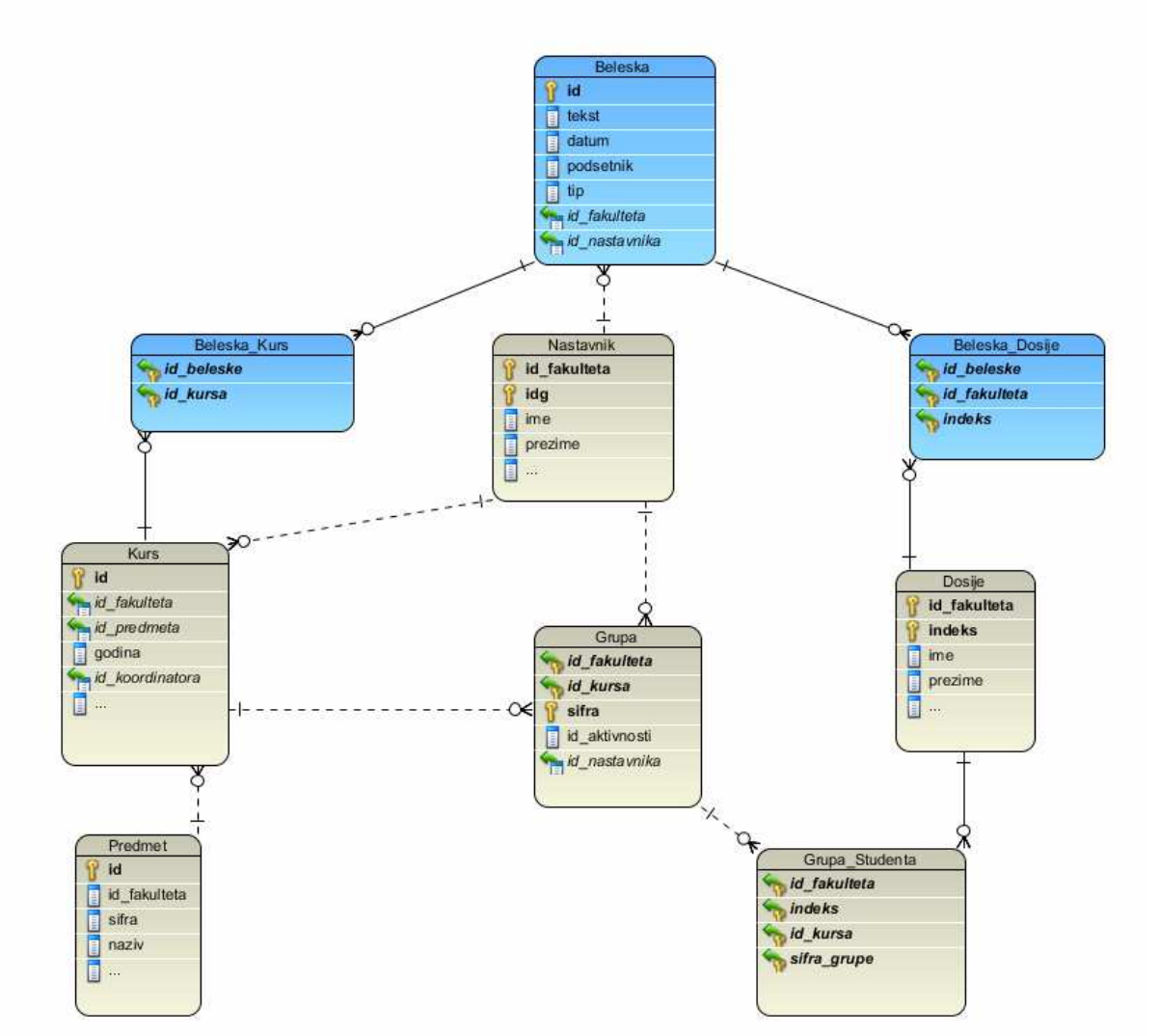
- naziv predmeta;
- šifra predmeta;
- fakultet na kome se predaje predmet.

Uočeno je da jedna beleška može biti vezana za više kurseva i da za jedan kurs može postojati više beleški. Kako je kardinalnost odnosa između entiteta ***Beleška*** i ***Kurs*** „više prema više", dodaje se asocijativni entitet ***Beleška_Kurs*** koji opisuje koja beleška se odnosi na koji kurs. Isto tako, jedna beleška može biti vezana za više studenata i za jednog studenta može biti postavljeno više beleški. Kardinalnost odnosa između entiteta ***Beleška*** i ***Dosije*** je „više prema više", stoga se dodaje asocijativni entitet ***Beleška_Dosije*** koji opisuje koja beleška se odnosi na kog studenta.

Kako jedna grupa može imati više studenata, a jedan student može biti raspoređen u više grupa, odnosno kardinalnost odnosa između entiteta ***Grupa*** i ***Dosije*** je „više prema više", stoga se dodaje se asocijativni entitet ***Grupa_Studenta***. Jedan student može biti u jednoj grupi iz jednog kursa u okviru jedne nastavne aktivnosti.

Dobijeni model entiteta i odnosa je grafički prikazan dijagramom entiteta i odnosa na slici 13. Nazivi entiteta i atributa su u skladu sa

postojećim neformalnim pravilima notacije u informacionom sistemu StudInfo. Radi preglednosti su izostavljeni atributi entiteta koji već postoje u sistemu StudInfo, a nisu značajni u kontekstu rada sa beleškama.



Slika 13: Dijagram entiteta i odnosa podsistema za nastavničke beleške

Analizom baze podataka informacionog sistema StudInfo uočeno je da već postoje tabele koje odgovaraju entitetima ***Kurs, Predmet, Dosije***, ***Nastavnik, Grupa*** i ***Grupa_Studenta*** sa informacijama potrebnim podsistemu za vođenje beleški.

Entiteti na dijagramu entiteta i odnosa prikazani sivom bojom već postoje u okviru informacionog sistema StudInfo, a entiteti obojeni plavom bojom su novi entiteti, koji su dodati radi potreba podsistema za vođenje beleški.

3.2 Shema baze podataka

Shema baze podataka je dobijena prevođenjem opisanih entiteta na dijagramu entiteta i odnosa u tabele. U nastavku je dat detaljan opis tako dibijenih tabela relacione baze podataka podsistema za nastavničke beleške.

Tabela	Opis
Beleska	Podaci o beleškama.
Nastavnik	Podaci o nastavku koji postavlja i pregleda beleške.
Dosije	Podaci o studentima za koje nastavnik postavlja beleške.
Kurs	Podaci o kursevima za koje nastavnik postavlja beleške.
Predmet	Podaci o predmetu koji se drži na kursevima za koje se postavljaju beleške.
Beleska_Kurs	Podaci koja beleška se odnosi na koje kurseve.
Beleska_Dosije	Podaci koja beleška se odnosi na koje studente.
Grupa	Podaci koji nastavnik učestvuje u održavanju kog tipa nastave na kom kursu, da li drži predavanja, vežbe ili praktikum.
Grupa_Studenti	Podaci koji student je u kojoj grupi.

U sledećim odeljcima je priložena struktura tabela baze podataka: nazivi kolona, tipovi podataka kolona, ograničenja ako postoje, opisi kolona i odnosi sa drugim tabelama. Za tabele koje su već postojale u informacionom sistemu StudInfo navedene su samo kolone koji se koriste u okviru podsistema za nastavničke beleške.

3.2.1 Beleska

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id	integer	PK	ne	Identifikator kursa
tekst	long vargraphic		ne	Tekst beleške
datum	timestamp		ne	Datum i vreme postavljanja ili poslednjeg menjanja beleške
podsetnik	smallint		ne	Oznaka da li je beleška postavljena kao podsetnik. Ima vrednost: 0 - „nije podsetnik“; 1 - „jeste podsetnik“. Podrazumevana vrednost je 0, odnosno „nije podsetnik“.
tip	char		ne	Oznaka da li je beleška postavljena kao lična ili službena. Ima vrednost: s - „javna beleška“; 1 – „lična beleška“. Podrazumevana vrednost je 1, odnosno „lična beleška“.
id_fakulteta	integer	SK	ne	Identifikator fakulteta na kojem se postavlja beleška.
id_nastavnika	integer	SK	ne	Identifikator nastavnika koji

	je uneo belešku.

Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Beleska	0..*	1	Nastavnik	Kolone (id_fakulteta, id_nastavnika) tabele Beleska predstavljaju strani ključ na tabelu Nastavik.
Beleska	1	0..*	Beleska_Kurs	Kolona (id_beleske) tabele Beleska_Kurs predstavlja strani ključ na tabelu Beleska.
Beleska	1	0..*	Beleska_Dosije	Kolona (id_beleske) tabele Beleska_Dosije predstavlja strani ključ na tabelu Beleska.

Ograničenja

Naziv	Implementacija	Opis
chk_tip	check(tip in ('l','s'))	Ograničenje da tip beleške može biti samo *l* (ako je lična beleška) ili *s* (ako je službena beleška).
chk_podsetnik	check(podsetnik in (0,1))	Ograničenje da podsetnik može biti samo *1* (ako je beleška podsetnik) ili *0* (ako beleška nije podsetnik).

3.2.2 Nastavnik

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id_fakulteta	integer	PK	ne	Identifikator fakulteta na kojem predaje nastavnik.
idg	integer	PK	ne	Identifikator člana nastavnog osoblja.
ime	vargraphic(50)		ne	Ime nastavnika
prezime	vargraphic(50)		ne	Prezime nastavnika
…	…	…	…	…

Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Nastavnik	1	0..*	Beleska	Kolone (id_fakulteta, id_nastavnika) tabele Beleska predstavljaju strani ključ na tabelu Nastavik.
Nastavnik	1	0..*	Grupa	Kolone (id_fakulteta, id_nastavnika) tabele Grupa predstavljaju strani ključ na tabelu Nastavi.k

3.2.3 Kurs

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id	integer	PK	ne	Identifikator beleške
id_fakulteta	integer	SK	ne	Identifikator fakulteta na kojem se odžava kurs.
id_predmeta	integer	SK	ne	Identifikator predmeta iz kojeg se drži kurs.
godina	smallint		ne	Školska godina u kojoj se održava kurs.
id_koordinatora	integer	SK	ne	Identifikator nastavnika koji je nadležan za organizaciju kursa.
...	...	...	...	...

Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Kurs	1	0..*	Beleska_Kurs	Kolona (id_beleske) tabele Beleska_Kurs predstavlja strani ključ na tabelu Beleska
Kurs	0..*	1	Predmet	Kolona (id_predmeta) tabele Kurs predstavlja strani ključ na tabelu Predmet
Kurs	1	0..*	Grupa	Kolone (id_fakulteta, id_kursa) tabele Grupa predstavljaju strani ključ na

				tabelu Kurs
Kurs	0..*	1	Nastavnik	Kolone (id_fakulteta, id_nastavnika) tabele Kurs predstavljaju strani ključ na tabelu Nastavnik

3.2.4 Predmet

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id	integer	PK	ne	Identifikator predmeta
id_fakulteta	integer		ne	Identifikator fakulteta na kojem se predaje predmet.
sifra	varchar(20)		ne	Šifra predmeta
naziv	varchar(200)		ne	Naziv predmeta
...	...	...	...	...

Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Predmet	1	0..*	Kurs	Kolona (id_predmeta) tabele Kurs predstavlja strani ključ na tabelu Predmet.

3.2.5 Beleska_Kurs

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id_beleske	integer	PK/SK	ne	Identifikator beleške
id_kursa	integer	PK/SK	ne	Identifikator kursa na koji se odnosi beleška.

Opis odnosa

Prva tabela	Druga tabela	Broj redova druge tabele pridružen jednom redu prve tabele	Broj redova prve tabele pridružen jednom redu druge tabele	Implementiran odnos
Beleska_Kurs	Beleska	1	0..*	Kolona (id_beleske) tabele Beleska_Kurs predstavlja strani ključ na tabelu Beleska.
Beleska_Kurs	Kurs	1	0..*	Kolona (id_beleske) tabele Beleska_Kurs

predstavlja strani ključ na tabelu Beleska.

3.2.6 Dosije

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id_fakulteta	integer	PK	ne	Identifikator fakulteta na kome student studira.
indeks	integer	PK	ne	Indeks studenta
ime	vargraphic(50)		ne	Ime studenta
prezime	vargraphic(50)		ne	Prezime studenta
…	…	…	…	…

Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Dosije	1	0..*	Grupa_Studenta	Kolone (id_fakulteta, indeks) tabele Grupa_Studenta predstavljaju strani ključ na tabelu Dosije.
Dosije	1	0..*	Beleska_ Dosije	Kolone (id_fakulteta, indeks) tabele Beleska_ Dosije predstavljaju strani ključ na tabelu Dosije.

3.2.7 Grupa_Studenta

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id_fakulteta	Integer	PK/SK	ne	Identifikator fakulteta na kome je grupa
indeks	Integer	PK/SK	ne	Indeks studenta koji je u grupi
id_kursa	Integer	PK/SK	ne	Identifikator kursa kome pripada grupa
sifra_grupe	vargraphic(20)	PK/SK	ne	Šifra grupe

Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Grupa_Studenta	0..*	1	Dosije	Kolone (id_fakulteta, id_indeksa) tabele Grupa_Studenta predstavljaju strani ključ na tabelu Dosije.
Grupa_Studenta	0..*	1	Grupa	Kolone (id_fakulteta, id_indeksa) tabele Grupa_Studenta predstavljaju strani ključ na tabelu Grupa.

3.2.8 Beleska_Dosije

Opis kolona

Kolona	Tip podatka	PK/SK	NULL vrednost	Opis
id_beleske	integer	PK/SK	ne	Identifikator beleške
id_fakulteta	integer	PK/SK	ne	Identifikator fakulteta na kome student studira.
indeks	indeks	PK/SK	ne	Indeks studenta na koga se odnosi beleška.

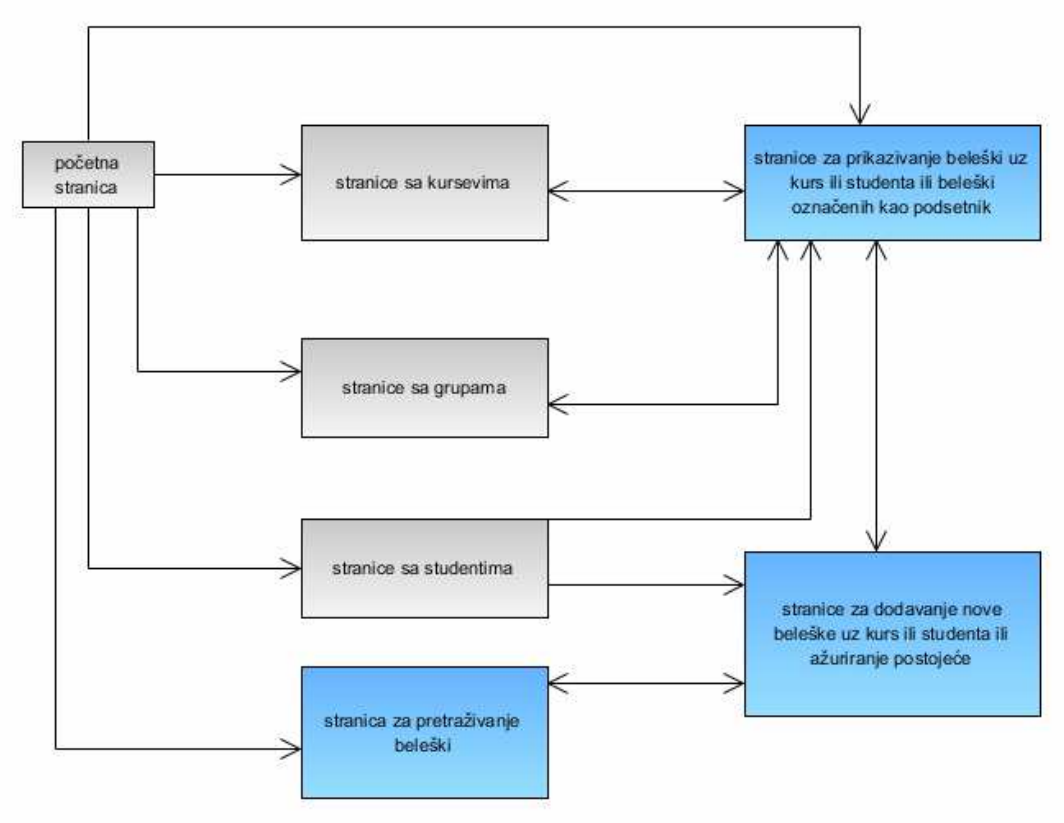
Opis odnosa

Prva tabela	Broj redova prve tabele pridružen jednom redu druge tabele	Broj redova druge tabele pridružen jednom redu prve tabele	Druga tabela	Implementiran odnos
Beleska_Dosije	0..*	1	Beleska	Kolona (id_beleske) tabele Beleska_Dosije predstavlja strani ključ na tabelu Beleska.
Beleska_Dosije	0..*	1	Dosije	Kolone (id_fakulteta, indeks) tabele Beleska_Dosije predstavljaju strani ključ na tabelu Dosije.

3.3 Korisnički interfejs

Kako podsistem za nastavničke beleške predstavlja deo informacionog sistema StudInfo, korisnički interfejs podsistema je oblikovan u skladu sa korisničkim interfejsom sistema StudInfo.

Povezanost stranica koje čine korisnički interfejs prikazani su dijagramom zavisnosti stranica u podsistemu za nastavničke beleške (slika 14).



Slika 14: Dijagram zavisnosti stranica u podsistemu za nastavničke beleške

Stranice na dijagramu zavisnosti stranica prikazane sivom bojom su već postojale u okviru informacionog sistema StudInfo, a stranice obojene plavom bojom su nove stranice, koje su dodate radi potreba podsistema za vođenje beleški. Postojeće stranice su izmenjene kako bi se nove stranice povezale sa njima i prikazivale neophodne podatke o beleškama.

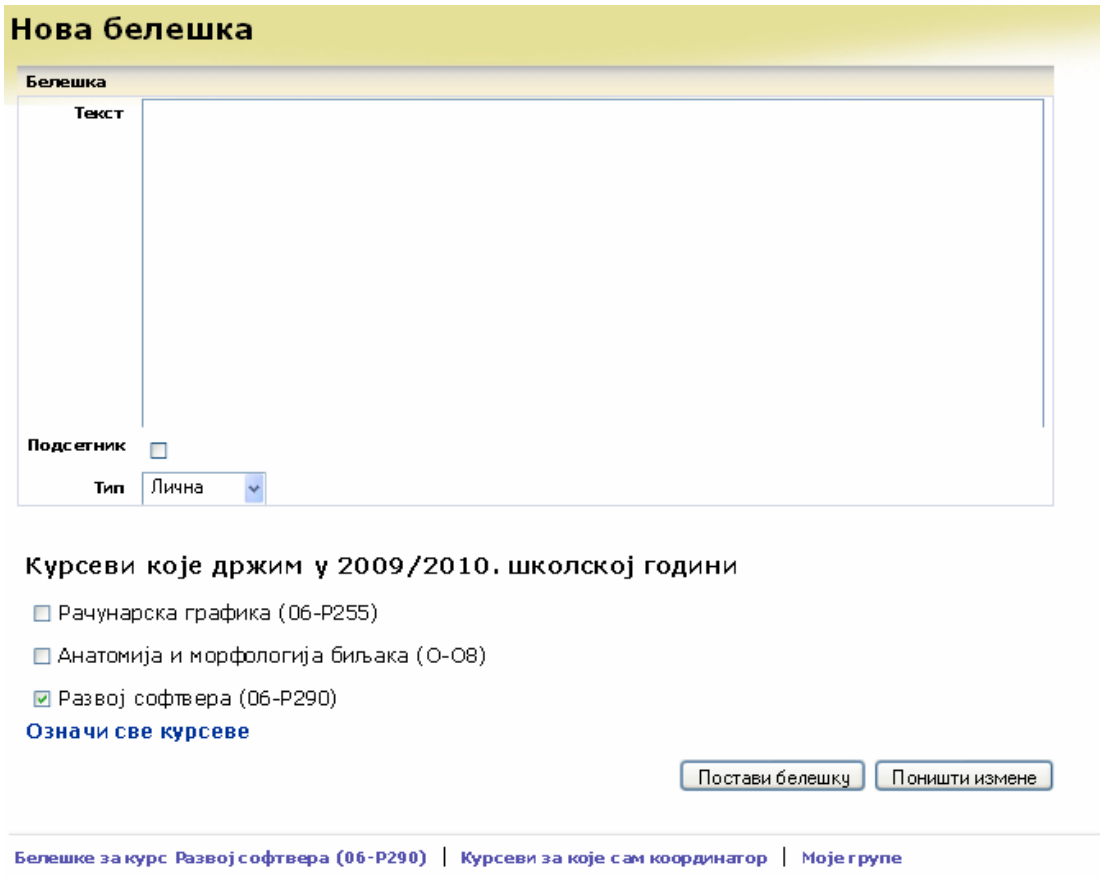
3.3.1 Dodavanje nove beleške uz kurs

Kada se na stranici za prikaz beleški uz neki kurs pritisne na vezu ***„Nova beleška“*** prikazuje se stranica sa formularom za unos podataka o novoj belešci uz kurseve (slika 15). Prikazuje se:

- tekstualno polje za unos teksta beleške;

- kvadratić za označavanje, preko koga se definiše da li je beleška podsetnik;
- padajuća lista za izbor tipa beleške, da li je beleška lična tj. moći će da je vidi samo nastavnik koji je postavio ili je službena tj. moći će da je vide svi nastavnici koji učestvuju u održavanju nastave na kursevima za koje će beleška biti postavljena;
- lista kurseva u čijoj nastavi učestvuje prijavljeni nastavnik. Svaki kurs je predstavljen jednim kvadratićem za označavanje koji treba da bude označen ukoliko beleška treba da bude postavljena za taj kurs.

Postoji i opcija *„**Označi sve kurseve**“* koja omogućava da se označe svi ponuđeni kursevi.



Slika 15: Unos nove beleške uz kurs

Postoje i dva dugmeta:

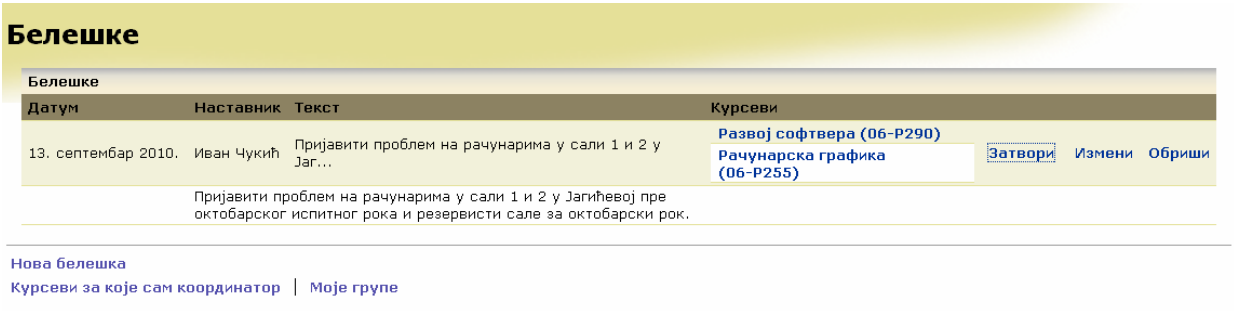
- *„**Postavi belešku**“* za postavljanje nove beleške i
- *„**Poništi izmene**“* za poništavanje izmena.

Prikazuju se i veze:

- *„**Prikaz beleški**“* koja vodi ka stranici za prikaz beleški uz kurseve;
- *„**Kursevi za koje sam koordinator**“* koja vodi ka stranici za prikaz kurseva za koje je prijavljeni nastavnik koordinator;
- *„**Moje grupe**“* koja vodi ka stranici za prikaz grupa prijavljenog nastavnika.

Kada se pritisne na dugme *„Postavi belešku“* ako nije unet tekst beleške ili nije izabran bar jedan kurs, prikazaće se prozorčić za poruke sa odgovarajućom porukom i beleška neće biti sačuvana.

Ako su svi neophodni podaci uneti, beleška će biti sačuvana i otvoriće se stranica za prikaz beleški na kojoj će se prikazati beleška koja je uneta (slika 16).



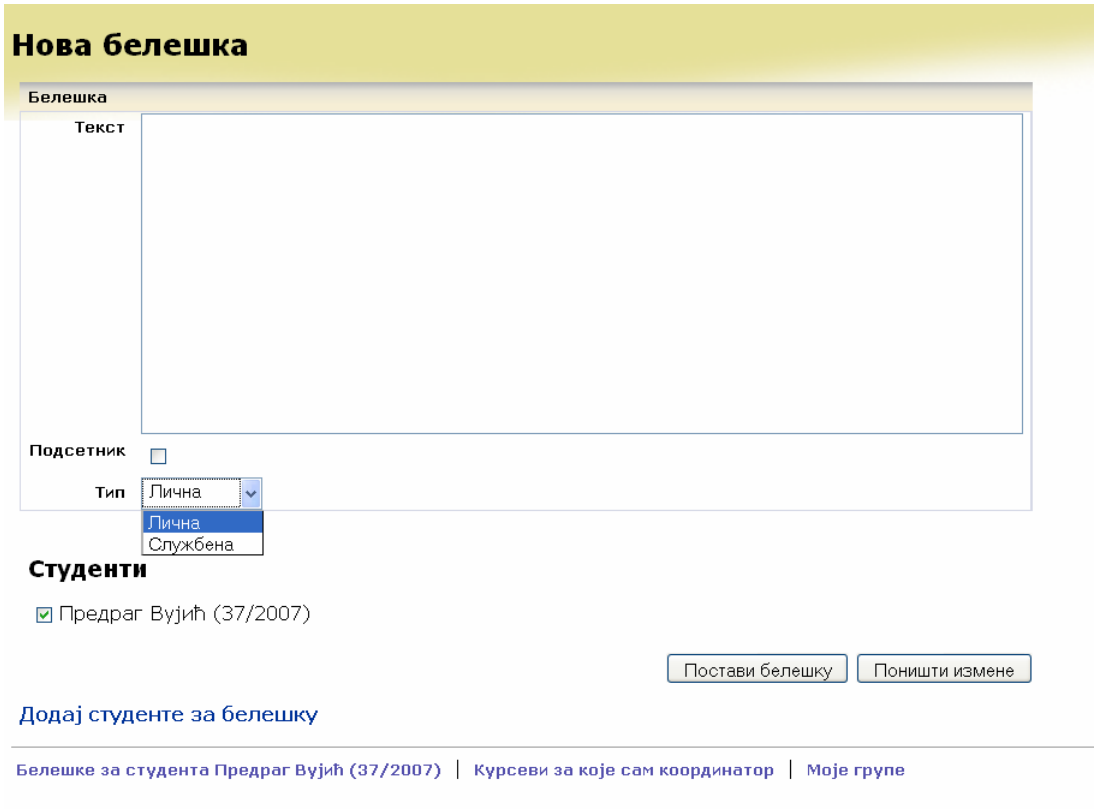
Slika 16: Prikaz sačuvane beleške uz kurseve

3.3.2 Dodavanje nove beleške uz studenta

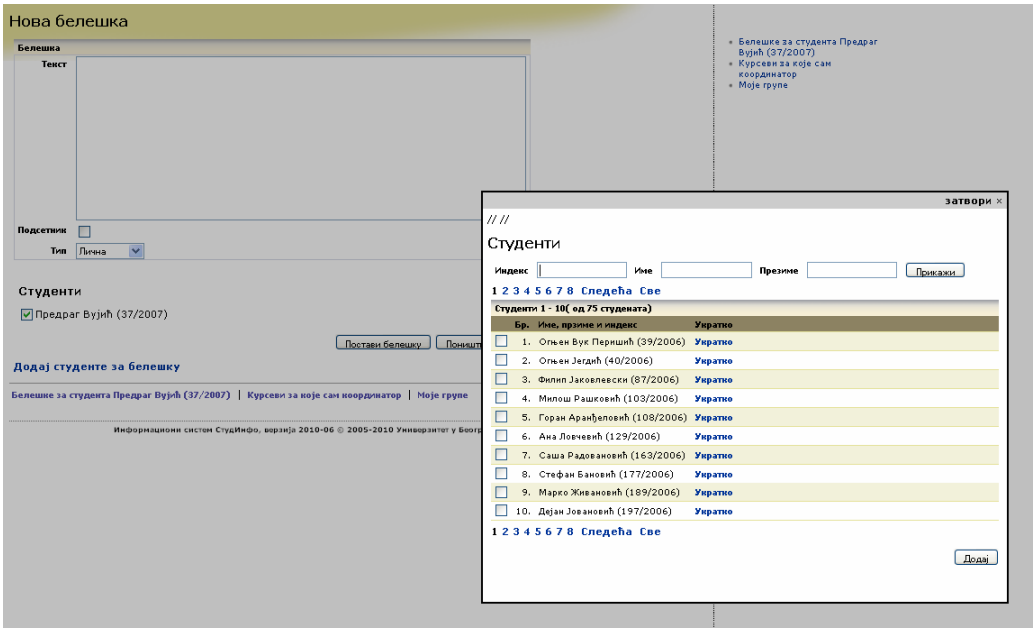
Kada se na stranici za prikaz beleški uz nekog studenta ili na stranici sa podacima o studentu pritisne na vezu *„Nova beleška“* prikazuje se stranica sa formularom za unos podataka o novoj belešci uz studente. Izgled stranice je isti kao i pri dodavanju nove beleške uz kurseve, osim što se umesto kurseva prikazuju studenti na koje će se odnositi beleška (slika 17).

Student koji je izabran na stranici za prikaz beleški je predstavljen pomoću kvadratića za označavanje koji je inicijalno označen.

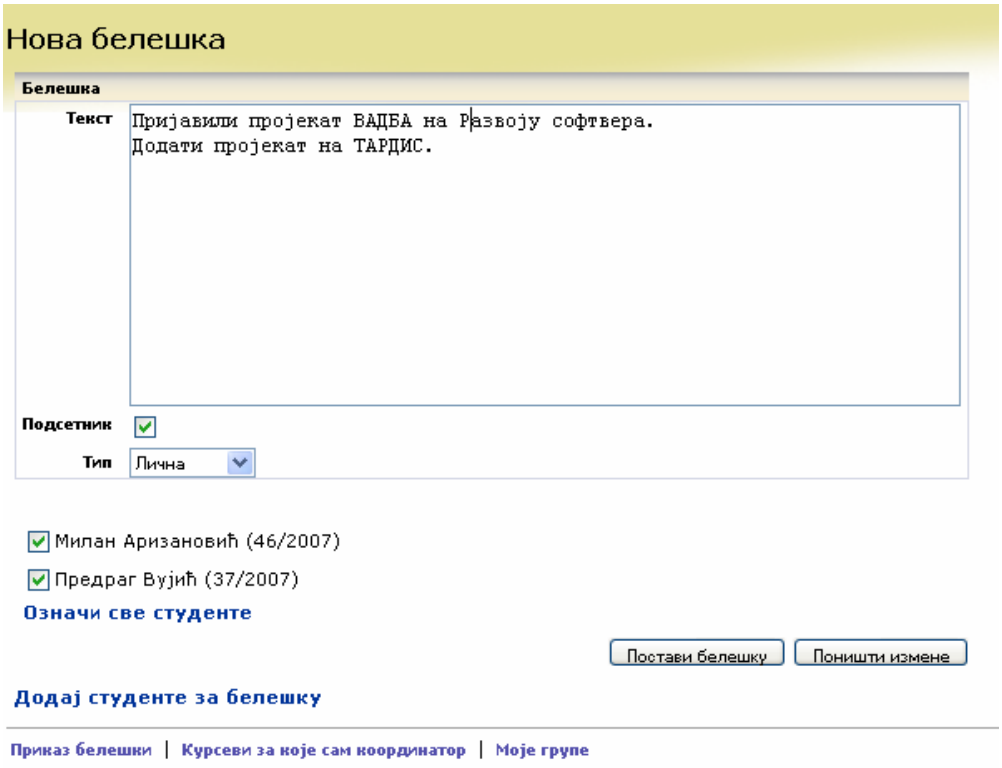
Postoji mogućnost da se doda još studenata na koje će se odnositi beleška. Pritiskom na vezu *„Dodaj studente za belešku“* prikazuje se prozorčić sa tabelom studenata kojima prijavljen nastavnik predaje u tekućoj školskoj godini, tj. spisak studenata za koje ima pravo da postavi beleške (slika 18). Za svakog studenta se prikazuje ime, prezime, indeks i veza *„Ukratko“* ka prozorčiću sa osnovnim podacima o studentu. U prozorčiću sa studentima omogućena je i pretraga studenata po indeksu, imenu ili prezimenu kako bi se olakšalo pronalaženje željenog studenta. Za svakog studenta je vezan kvadratić za označavanje koji nastavnik treba da označi ukoliko želi da se beleška odnosi na studenta. Pritiskom na dugme *„Dodaj”* izabrani studenti se dodaju prethodno izabranim studentima na koje će se odnositi beleška. Svaki od studenata je predstavljen kvadratićem za označavanje kojim se određuje da li će se beleška odnositi na njih. Postoji mogućnost da se nastavnik predomisli, da ne želi da se beleška odnosi na nekog prethodno izabranog studenta, u tom slučaju ne treba da bude označen pridružen kvadratić (slika 19).



Slika 17: Unos nove beleške uz studenta



Slika 18: Dodavanje novih studenata

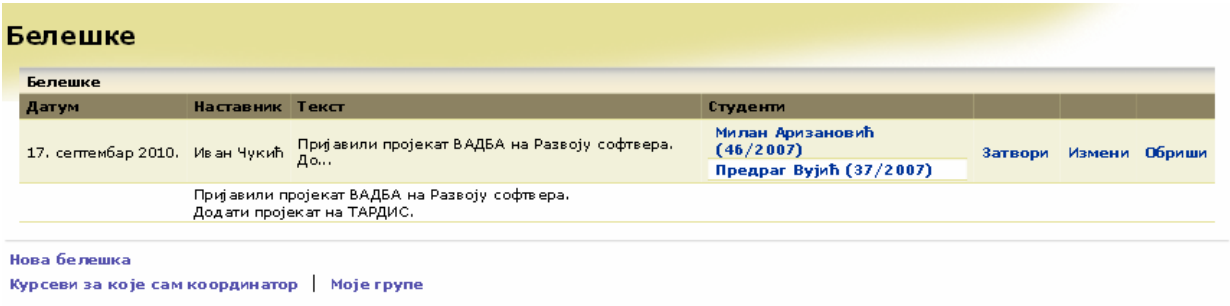


Slika 19: Prikaz dodatih studenata za belešku

Kada postoji više od jednog studenta na koje se odnosi beleška, umesto veze „***Označi sve kurseve***“ prikazuje se veza „***Označi sve studente***“, koja obezbeđuje da svi studenti koji su prikazani na stranici sa podacima o novoj belešci budu izabrani.

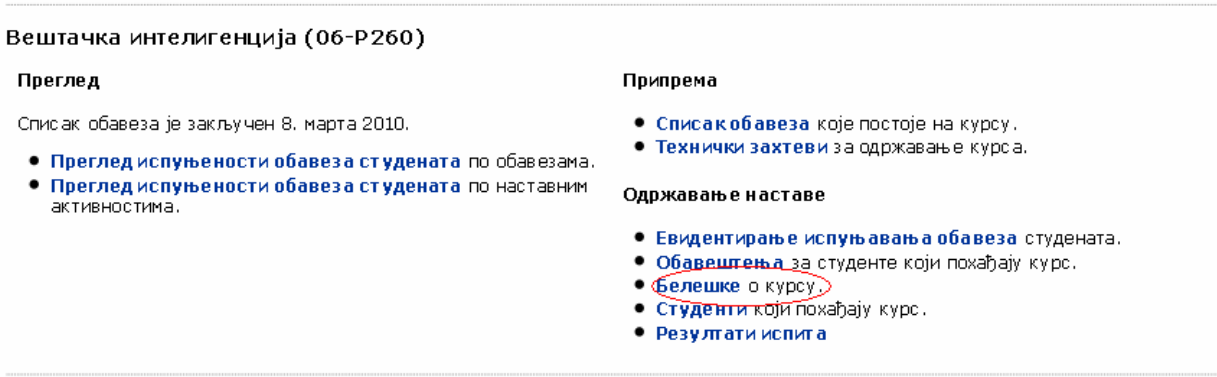
Kada se pritisne dugme „***Postavi belešku***“ proverava se da li je unet tekst beleške i da li je izabran bar jedan student na kojeg će se odnositi beleška. Ako neki od uslova nije ispunjen prikazuje se odgovarajuća poruka i beleška neće biti sačuvana dok se ti uslovi ne ispune.

Ako su svi neophodni podaci uneti, beleška će biti sačuvana i otvoriće se stranica za prikaz beleški na kojoj će se prikazati beleška koja je uneta uz studente (slika 20).



Slika 20: Prikaz nove beleške uz studente

3.3.3 Pregledanje beleški uz kurs



Slika 21: Deo stranice sa podacima o kursevima

Postojećim stranicama sistema StudInfo, koje se odnose na kurseve i grupe su dodate veze ka podsistemu za nastavničke beleške.

Na stranici koja je posvećena kursevima (slika 21) čiji je koordinator prijavljeni nastavnik, prikazuje se spisak kurseva. Uz svaki kurs je naveden spisak uobičajenih aktivnosti koje se odnose na taj kurs. U grupi aktivnosti „***Održavanje nastave***“ dodaje se veza „***Beleške o kursu***“ (zaokružena na slici) koja vodi ka stranici na kojoj se prikazuju beleške koje je nastavnik postavio uz taj kurs, kao i službene beleške.

Pošto saradnici u nastavi nisu koordinatori ni za jedan kurs, oni nemaju stranicu posvećenu kursevima. Zbog toga je na stranici koja je posvećena grupama, uz informacije o svakoj grupi dodata veza „***Pregled beleški uz kurs***“. Veza vodi ka stranici na kojoj se prikazuju beleške uz kurs kome pripada grupa.

Korisnički interfejs stranica posvećenih kursevima i grupama kurseva je projektovao razvojni tim sistema StudInfo. Postojećim stranicama su dodate veze ka podsistemu za nastavničke beleške.

Na stranici koja sadrži pregled beleški uz kurs (slika 22) prikazuje se tabela u kojoj se za svaku belešku prikazuju podaci u dva reda. U prvom redu se prikazuju osnovni podaci o belešci. U drugom redu se prikazuje celokupan tekst beleške. Tekst beleške se inicijalno ne prikazuje radi preglednosti. Prikazuju se beleške koje je postavio prijavljeni nastavnik i sve službene beleške.

U prvom redu se prikazuje:

- datum postavljanja beleške (ili poslednje promene, ukoliko je nastavnik ažurirao datum postavljanja);
- ime i prezime nastavnika koji je postavio belešku;
- prvih 50 znakova teksta beleške;

- lista svih kurseva na koje se odnosi beleška. Svaki kurs je predstavljen kao veza ka stranici na kojoj se prikazuju beleške koje su postavljene uz taj kurs.
- ako nije prikazan tekst beleške prikazuje se veza „***Otvori***“ koja omogućava da se prikaže pun tekst beleške u redu ispod reda sa podacima o belešci. U suprotnom je prikazana veza „***Zatvori***“ koja omogućava da se sakrije red sa tekstom beleške;
- ukoliko je belešku koja se prikazuje postavio prijavljeni nastavnik u toku tekuće školske godine, prikazuje se veza „***Izmeni***“ koja vodi ka stranici na kojoj je moguće ažurirati podatke o belešci;
- ukoliko je belešku koja se prikazuje postavio prijavljeni nastavnik u toku tekuće školske godine, prikazuje se veza „***Obriši***“ koja omogućava brisanje beleške. Kada se klikne na vezu prikazuje se poruka koja zahteva potvrdu brisanja pre nego što se beleška obriše.

Slika 22: Prikaz beleški uz kurs

U vrhu stranice za prikazivanje beleški postoji padajuća lista „***Kurs***“ koja sadrži sve kurseve u čijoj nastavi u tekućoj školskoj godini učestvuje prijavljeni nastavnik. Kada se odabere kurs iz padajuće liste, menja se sadržaj stranice tako da se prikazuju beleške pridružene izabranom kursu.

Ispod padajuće liste „***Kurs***“ prikazuje se padajuća lista „***Godine***“. Ona sadrži sve školske godine u kojima je prijavljeni nastavnik učestvovao u održavanju nastave iz predmeta koji se drži na kursu izabranom preko liste „***Kurs***“. Odabirom školske godine prikazaće se beleške koje su postavljene uz kurs iz istog predmeta koji je održavan u izabranoj školskoj godini.

Prikazuju se i veze:

- ***„Nova beleška“*** koja vodi ka stranici za dodavanje nove beleške uz kurseve, ako se beleške koje se prikazuju odnose na kurs iz tekuće školske godine;
- ***„Kursevi za koje sam koordinator“*** koja vodi ka stranici za prikaz kurseva čiji je koordinator prijavljeni nastavnik;

- ***„Moje grupe“*** koja vodi ka stranici za prikaz grupa prijavljenog nastavnika.

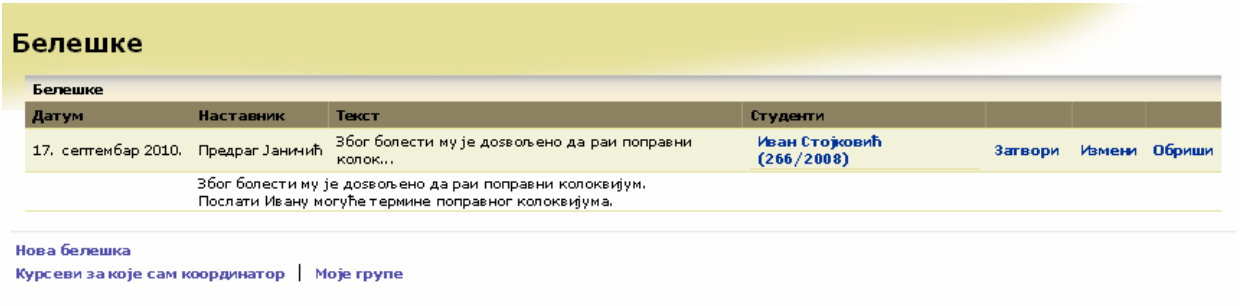
3.3.4 Pregledanje beleški uz studenta

U okviru informacionog sistema StudInfo projektovan je korisnički interfejs za stranicu sa osnovnim podacima o svakom studentu. Radi potreba podsistema za nastavničke beleške stranici se dodaje veza ***„Beleške o studentu“*** ukoliko postoji beleška o studentu za koju prijavljeni nastavnik ima prava da je vidi (slika 23). Nastavnik može da vidi sve službene beleške uz studente ili beleške koje je on postavio uz studenta kome predaje.

Kada se klikne na vezu ***„Beleške o studentu“*** prikazuje se stranica za pregledanje beleški postavljenih uz studenta. Stranica za pregledanje beleški uz studenta je slična kao stranica za pregledanje beleški uz kurs. Umesto kurseva na koje se odnosi beleška prikazani su studenti na koje se odnosi beleška. Na stranici za pregledanje beleški uz studente nema padajućih lista za izbor kurseva i školskih godina (slika 24). Ako želi da pregleda beleške za drugog studenta, nastavnik ga mora potražiti kroz postojeći korisnički interfejs za pretraživanje studenata.

Ako ne postoje beleške uz studenta koje prijavljeni nastavnik može da vidi, a ima pravo da postavi novu belešku, na stranici o studentu se umesto veze „***Beleške o studentu***“ prikazuje veza ***„Nova beleška“*** koja vodi ka stranici za unos nove beleške uz tog studenta.

Slika 23: Prikaz podataka o studentu



Slika 24: Prikaz beleški uz studenta

3.3.5 Ažuriranje beleške uz kurseve

Kada se na nekoj od stranica koja prikazuje beleške uz kurseve izabere mogućnost ažuriranja beleške, otvara se stranica za unos beleške. Sadržaj formulara je inicijalno popunjen postojećim podacima o izabranoj belešci i može se menjati (slika 25).

Umesto dugmeta „***Postavi belešku***“ prikazuje se dugme „***Promeni belešku***“ za čuvanje promenjenih podataka beleške.

Sve ostalo je isto kao u slučaju dodavanja nove beleške.

Nakon ažuriranja beleške, izmenjena beleška se prikazuje u okviru stranice za prikaz beleški.

Slika 25: Ažuriranje beleške uz kurseve

3.3.6 Ažuriranje beleške uz studente

Kada se na nekoj od stranica koja prikazuje beleške uz studente izabere mogućnost ažuriranja beleške, otvara se stranica za unos beleške. Sadržaj formulara je inicijalno popunjen postojećim podacima o izabranoj belešci i može se menjati (slika 26).

Umesto dugmeta „***Postavi belešku***“ prikazuje se dugme „***Promeni belešku***“ za čuvanje promenjenih podataka beleške.

Sve ostalo je isto kao u slučaju dodavanja nove beleške. Moguće je i dodavati nove studente na isti način kao i pri dodavanju nove beleške.

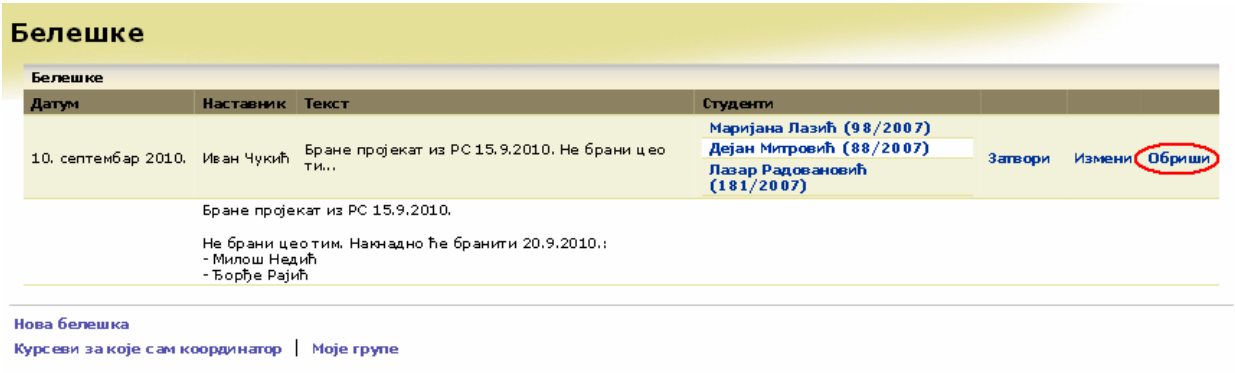
Kada se sačuva beleška, izmenjena beleška se prikazuju na stranici za prikaz beleški.

Slika 26: Ažuriranje beleške uz studente

3.3.7 Brisanje postojeće beleške

Na stranicama za prikazivanje beleški uz kurseve iz tekuće školske godine ili studente uz svaku belešku koju je postavio prijavljeni nastavnik postoji opcija „***Obriši***“ za brisanje beleške iz baze podataka (slika 27). Kada se izabere opcija za brisanje beleške prikazuje se prozorčić sa porukom koji

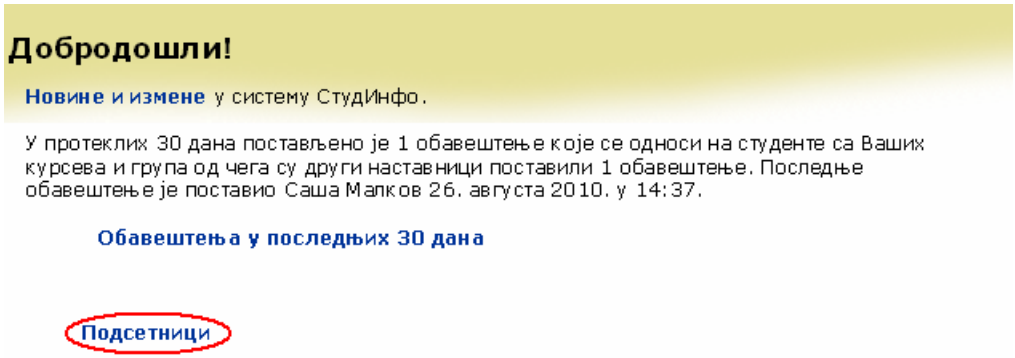
zahteva da se potvrdi zahtev za brisanjem beleške. Ako nastavnik potvrdi zahtev beleška će biti obrisana iz baze podataka.



Slika 27: Brisanje beleške

3.3.8 Pregledanje beleški označenih kao podsetnika

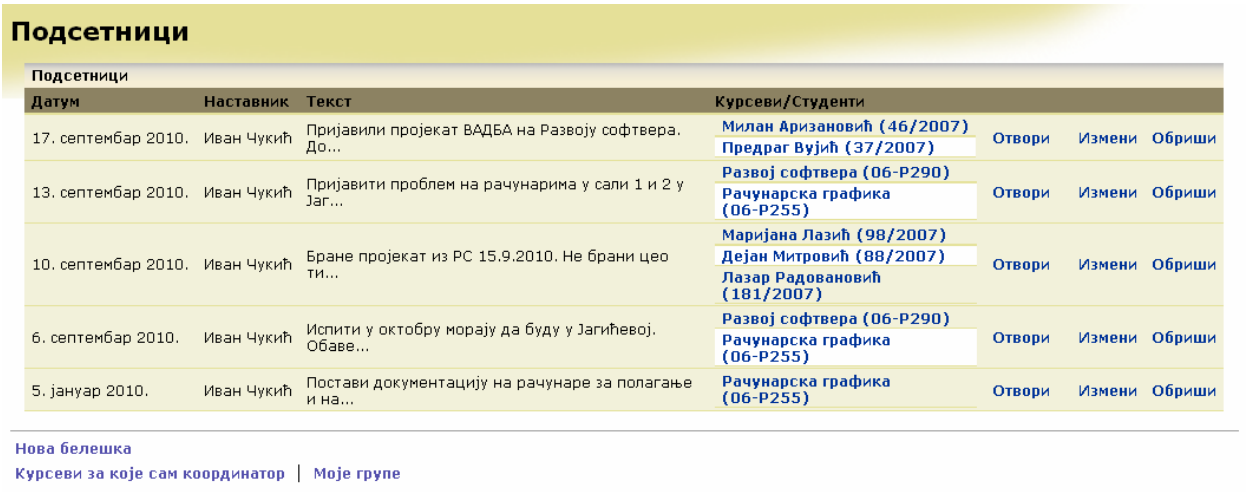
Ukoliko postoji bar jedna beleška označena kao podsetnik koju nastavnik ima pravo da vidi, na početnoj stranici će postojati veza „***Podsetnici***“ ka stranici za pregledanje beleški označenih kao podsetnici (slika 15).



Slika 28 : Poruka o podsetnicima na početnoj strani

Izgled stranice za pregledanje beleški označenih kao podsetnici je sličan kao kod stranice za pregledanje beleški uz kurs ili studenta. Prikazuju se sve beleške uz kurseve i studente koje su označene kao podsetnici. U okviru podataka o belešci se prikazuje lista kurseva ili studenata, zavisno od toga uz šta je postavljena beleška.

Veza ***„Nova beleška“*** vodi ka stranici za dodavanje nove beleške uz kurs. Veze „***Izmeni***“ i „***Obriši***“ omogućavaju menjanje i brisanje postojećih beleški.



Slika 28: Prikaz podsetnika

3.3.9 Pretraživanje beleški uz kurseve i studente

Iz glavnog menija aplikacije (slika 29) moguće je izabrati opciju „***Beleške***“ koja vodi ka stranici za pretraživanje beleški po različitim kriterijumima (slika 30).

Slika 29: Izbor pretraživanja beleški

Preko padajuće liste „***Tip pretrage***“ se bira da li se pretraga vrši po:

- svim beleškama – obuhvata beleške koje su postavljene uz kurseve i beleške koje su postavljene uz studente;
- beleškama koje su postavljene samo uz kurseve ili
- beleškama koje su postavljene samo uz studente.

Za sve tipove pretrage moguće je odabrati period postavljanja beleške, tip beleške i navesti deo teksta beleške.

Period postavljanja beleške se bira iz padajuće liste. Dopuštene opcije su:

- u poslednjih 30 dana;
- u toku tekuće školske godine ili
- u poslednjih pet godina.

Ako je izabrana opcija poslednjih pet godina i pretraga uključuje kurseve tada se pretraga vrši po predmetima. Ako je izabran jedan kurs, traže se beleške koje su se vodile uz kurseve u poslednjih pet godina iz predmeta koji se sluša na izabranom kursu. Ako je izabrana opcija pretrage po svim kursevima onda se traže beleške koje su se vodile uz kurseve iz svih predmeta koje nastavnik predaje u tekućoj školskoj godini.

Tip beleške se bira preko padajuće liste. Dopuštene opcije su:

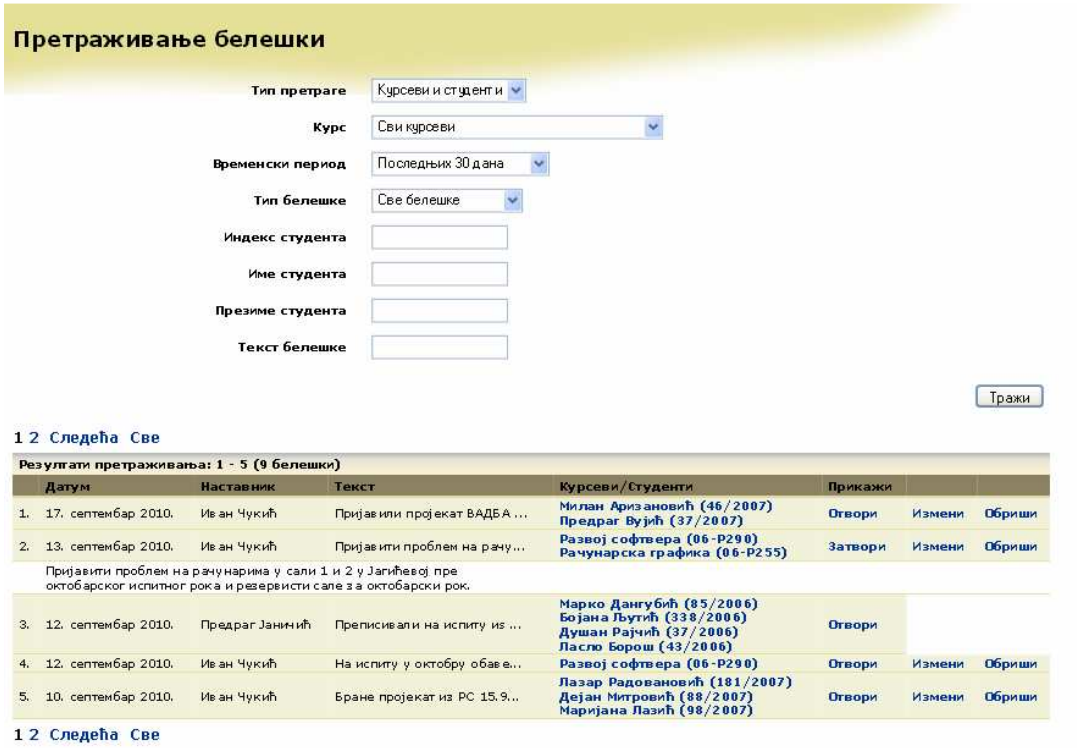
- lične beleške – samo beleške koje je postavio prijavljeni nastavnik;
- službene beleške – samo beleške koje je postavio drugi nastavnik, a prijavljeni nastavnik ima pravo da ih vidi;
- lične i službene beleške.

Ako se navede reč ili fraza iz teksta beleške, tada će se pronaći samo one beleške koje doslovno sadrže zadati tekst.

Ako pretraga uključuje beleške uz kurseve, pomoću padajuće liste „***Kurs***“ je moguće izabrati kurs za koji se vrši pretraga. Među dopuštenim opcijama se nalaze kursevi koje nastavnik drži u tekućoj školskoj godini i postoji opcija „*Svi kursevi*“ koja definiše da se pretraga odnosi na sve kurseve.

Ako pretraga uključuje beleške uz studente moguće je pretragu suziti prema:

- indeksu studenta;
- imenu studenta - može se uneti i samo deo imena;
- prezimenu studenta - može se uneti i samo deo prezimena.



Slika 30: Pretraživanje beleški uz kurseve i studente

Beleške koje predstavljaju rezultat pretrage prikazuju se u tabeli u kojoj se za svaku belešku rezultata prikazuju podaci u dva reda. U prvom redu se prikazuju osnovni podaci o belešci, a u drugom redu se prikazuje celokupan tekst beleške. Tekst beleške se inicijalno ne prikazuje. Za svaku belešku se prikazuje:

- datum postavljanja beleške;
- ime i prezime nastavnika koji je postavio belešku;
- prvih 50 znakova teksta beleške;
- lista svih studenata ili kurseva na koje se odnosi beleška. Svaki student ili kurs je predstavljen kao veza ka stranici na kojoj se prikazuju beleške koje su postavljene uz tog studenta ili kurs.
- veza „***Otvori***“ ili „***Zatvori***“ zavisno od toga da li je prikazan red sa tekstom beleške. Ako nije prikazan tekst beleške prikazuje se veza „***Otvori***“ koja omogućava da se prikaže pun tekst beleške u redu ispod reda sa podacima o belešci, u suprotnom se prikazuje veza „***Zatvori***“ koja omogućava da se ne prikazuje red sa tekstom beleške.
- ukoliko je belešku koja se prikazuje postavio prijavljeni nastavnik prikazuju se veze „***Izmeni***“ i „***Obriši***“ koje omogućavaju menjanje i brisanje beleške.

Ukoliko ne postoji beleška koja zadovoljava kriterijum pretrage prikazuje se odgovarajuća poruka.

4 Implementacija

4.1 Baza podataka

Naredbe za pravljenje tabela ***Beleska***, ***Beleska_Kurs*** i ***Beleska_Dosije*** su najpre napravljene automatski, na osnovu dijagrama entiteta i odnosa, pomoću alata *Database Visual ARCHITECT.* Dobijene naredbe su zatim manuelno prilagođene relacionom sistemu *IBM DB2* i načinu organizacije podataka u sistemu StudInfo.

```
CREATE TABLE dbn.Beleska (
   id integer NOT NULL,
   tekst long vargraphic NOT NULL,
   datum timestamp DEFAULT current timestamp NOT NULL,
   podsetnik smallint DEFAULT 0 NOT NULL,
   id_fakulteta integer NOT NULL,
   tip char,
   id_nastavnika integer NOT NULL,

   PRIMARY KEY (id),

   FOREIGN KEY FKBeleska_Nast(id_fakulteta, id_nastavnika)
      REFERENCES
           dbn.Nastavnik (id_fakulteta, idg),

   constraint chk_tip check( tip in ('l','s') ),
   constraint chk_podsetnik check( podsetnik in (0,1) )
)
in userspace1
index in index_4k
long in longdata
;

CREATE TABLE dbn.Beleska_Kurs (
   id_beleske integer NOT NULL,
   id_kursa integer NOT NULL,

   PRIMARY KEY (id_beleske, id_kursa),

   FOREIGN KEY FKBeleska_Kurs1 (id_beleske)
      REFERENCES
           dbn.Beleska (id),
   FOREIGN KEY FKBeleska_Kurs2(id_kursa)
      REFERENCES
           dbn.Kurs (id)
)
in userspace1
index in index_4k
long in longdata
;

CREATE TABLE dbn.Beleska_Dosije (
   id_beleske integer NOT NULL,
   id_fakulteta integer NOT NULL,
   indeks integer NOT NULL,
```

```
    PRIMARY KEY (id_beleske, id_fakulteta, indeks),

    FOREIGN KEY FKBeleska_Dosije1(id_beleske)
      REFERENCES
           dbn.Beleska (id),
    FOREIGN KEY FKBeleska_Dosije2(id_fakulteta, indeks)
      REFERENCES
           dbn.Dosije (id_fakulteta, indeks)
)
in userspace1
index in index_4k
long in longdata
;
```

Slika 31: Naredbe za pravljenje tabela podsistema za nastavničke beleške

4.2 Klase podataka

Da bi podaci iz tabela relacione baze podataka mogli da se koriste u programskom jeziku *Java* potrebno je implementirati klase pomoću kojih bi se predstavljali u objektno orjentisanim aplikacijama. Takve klase se obično nazivaju *stalne* (eng. *persistent*) klase. *Hibernate* zahteva da stalne klase koje se koriste za preslikavanje implemetiraju određena pravila koja zadovoljavaju tzv. *POJO (*eng. *plain old Java objects*) ili *proste* klase.

POJO klase implementiraju:

- atribute koji odgovaraju kolonama u tabeli;
- metode za pristup atributima. Za svaki atribut postoji metod *get* koji vraća vrednost atributa i metod *set* koji postavlja vrednost atributa. Ime metoda get se formira tako što se iza reči *get* dodaje ime atributa. Po istom principu se formira i ime metode *set*.
- konstruktor bez argumenata;
- konstruktor sa atributima koji odgovaraju kolonama u tabeli koje ne smeju da imaju nedefinisane vrednosti;
- konstruktor sa svim atributima;
- atribut koji predstavlja skup objekata druge klase. Ukoliko je jednom objektu *POJO* klase pridruženo više objekata druge *POJO* klase dodaje se atribut tipa *java.util.Set* ili *java.util.List* i predstavlja skup objekata druge klase.
- interfejs *java.io.Serializable*, čiju implementaciju ne zahteva *Hibernate*, ali ga je neophodno implemetirati da bi se objekti mogli koristiti u sesijama.

Za početnu implementaciju klasa u programskom jeziku *Java* koje odgovaraju tabelama u relacionoj bazi podataka korišćen je alat *hbm2java* koji na osnovu *SQL* sheme generiše *POJO* klase.

Na taj način su napravljene klase: ***Beleska***, ***BeleskaKurs***, ***BeleskaDosije***. Njihovi atributi su definisani na osnovu kolona u tabelama ***Beleska***, ***Beleska_Kurs***, ***Beleska_Dosije***. Klase ***Beleska*** i ***BeleskaKurs*** su

navedene u dodatku A. Kako je klasa ***BeleskaDosije*** veoma slična klasi ***BeleskaKurs***, ona nije priložena.

Jednom objektu klase ***Beleska*** može da se pridruži jedan ili više objekata klase ***BeleskaKurs***. Odnos između beleške i kurseva je opisan pomoću skupa objekata klase ***BeleskaKurs***, koji je dodat kao atribut klasi ***Beleska.***

Na isti način je implementiran i odnos između objekta klase ***Beleska*** i objekata klase ***BeleskaDosije.***

U tabelama relacione baze podataka entiteti koji se opisuju se identifikuju pomoću primarnog ključa, a u programskom jeziku *Java* objekti iste klase se razlikuju korišćenjem metoda *equals* i *hashCode.* Da bi se obezbedilo da se objekti u programskom jeziku *Java* porede na osnovu vrednosti atributa koji odgovaraju kolonama primarnog ključa u tabeli u kojoj se čuvaju, potrebno je predefinisati metode *equals* i *hashCode* tako da se poređenje vrši na osnovu preslikanih vrednosti primarnog ključa. Stoga alat *hbm2java* za svaku tabelu koja ima složen primarni ključ pravi po jednu *Java* klasu koja služi za pravljenje identifikatora objekta. Klase koje se koriste za identifikaciju objekata moraju da sadrže metode *equals* i *hashCode* za poređenje objekata.

Pošto tabela ***Beleska_Kurs*** ima složen primarni ključ, koji čine kolone ***id_beleske*** i ***id_kursa***, alat *hbm2java* pored klase ***BeleskaKurs*** generiše i klasu ***BeleskaKursId*** kojom se opisuje identifikator objekata klase ***BeleskaKurs.*** Identifikator se koristi za upoređivanje dva objekta klase ***BeleskaKurs*** . U dodatku A je navedena i klasa ***BeleskaKursId.***

Na sličan način je generisana i klasa ***BeleskaDosijeId*** čiji objekat predstavlja identifikator objekta klase ***BeleskaDosije***.

Nijedna od *POJO* klasa nije menjana nakon automatskog pravljenja.

Pored *POJO* klasa koje predstavljaju preslikavanje tabela iz relacione baze podataka ili preslikavanje primarnih ključeva tabela, za svaku *POJO* klasu pravi se po jedna pomoćna *Home* klasa[1] koja je zadužena za pristup bazi radi traženja objekata i za izvršavanje nekih operacija nad njima. *Home* klase mogu da sadrže pomoćne metode koje pristupaju bazi, izvršavaju upite i vraćaju rezultat upita, komparatore za poređenje objekata *POJO* klasa i druge pomoćne metode ili klase.

Nakon automatskog pravljenja klasa ***BeleskaHome*** je izgledala kao u prilogu (slika 32).

---

[1] Pomoćne klase dobijaju ime tako što se imenu klase kojoj se pridružuje dodaje reč *Home.* Na primer, za *POJO* klasu *Beleska* pravi se klasa *BeleskaHome.* Zbog toga se nazivaju *Home* klase. Njihova osnovna namena je da se sve manuelne nadgradnje ponašanja izmeste iz POJO klase, kako bi se ona po potrebi mogla lako automatski prilagođavati promenama strukture baze podataka.

```
package matf.studinfo.db.home;

import matf.studinfo.db.*;
import matf.studinfo.db.utils.*;

public class BeleskaHome extends GenericHibernateHome<Beleska, int>
{
   public BeleskaHome()
   {
      super(Beleska.class);
   }
}
```

Slika 32: Izgled automatski generisane klase BeleskaHome

Tokom implementacije podsistema za nastavničke beleške, klasi ***BeleskaHome*** su dodate neophodne metode za pretraživanje baze podataka i komparator za poređenje objekata klase ***Beleska*** prema datumu postavke beleški. U dodatku A je navedena cela klasa ***BeleskaHome***, u dovršenom obliku.

4.3 Objektno-relaciono preslikavanje

Pored *Java* klasa, alat *hbm2java* na osnovu *SQL* sheme pravi *XML* datoteke koje opisuju preslikavanje između napravljenih klasa i tabela. U datotekama preslikavanja za *Hibernate* definiše se koja klasa u sistemu odgovara kojoj tabeli. Preslikavanje se navodi kao sadržaj etikete <*hibernate-mapping*>. Za svaku klasu preslikavanje se opisuje pomoću etiketa:

- *class* - opisuje koja *POJO* klasa se koristi za preslikavanje koje tabele u bazi podataka i u kojoj shemi se nalazi tabela;
- *id* - opisuje koji atribut u klasi predstavlja identifikator objekta klase, kako se pravi i kog je tipa. Identifikator objekta klase odgovara primarnom ključu u tabeli. Ukoliko je primarni ključ složen koristi se etiketa *composite-id* za opis koja klasa se koristi za predstavljanje identifikatora objekta i koji atributi te klase odgovaraju kojoj koloni primarnog ključa.
- *property* - deklariše atribute klase, njihove tipove i njima odgovarajuće kolone tabele. Za svaki atribut postoji po jedna etiketa *property*;
- *set* - opisuje skup objekata neke klase koji se pridružuju jednom objektu klase koja se opisuje;
- *many-to-one* i *one-to-many* - opisuju odnose sa objektima drugih klasa.

Na slici (slika 33) je kao primer preslikavanja naveden sadržaj datoteke *Beleska.hbm.xml* za tabelu ***Beleska***.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="matf.studinfo.db.Beleska" table="BELESKA" schema="DBN">
        <id name="id" type="int">
            <column name="ID" />
            <generator class="assigned" />
        </id>
        <many-to-one name="nastavnik" class="matf.studinfo.db.Nastavnik"
  fetch="select">
            <column name="ID_FAKULTETA" not-null="true" />
            <column name="ID_NASTAVNIKA" not-null="true" />
        </many-to-one>
        <property name="tekst" type="string">
            <column name="TEKST" length="32700" not-null="true" />
        </property>
        <property name="datum" type="timestamp">
            <column name="DATUM" length="26" not-null="true" />
        </property>
        <property name="podsetnik" type="short">
            <column name="PODSETNIK" not-null="true" />
        </property>
        <property name="tip" type="char">
            <column name="TIP" length="1" not-null="true" />
        </property>
        <set name="beleskaDosijes" inverse="true">
            <key>
                <column name="ID_BELESKE" not-null="true" />
            </key>
            <one-to-many class="matf.studinfo.db.BeleskaDosije" />
        </set>
        <set name="beleskaKurses" inverse="true">
            <key>
                <column name="ID_BELESKE" not-null="true" />
            </key>
            <one-to-many class="matf.studinfo.db.BeleskaKurs" />
        </set>
    </class>
</hibernate-mapping>
```

Slika 33: Datoteka za preslikavanje Beleska.hbm.xml

U datoteci je definisano da se identifikator beleške (tj. njen primarni ključ) pravi automatski. Definiše se i odnos „više prema jedan“ između klasa ***Beleska*** i ***Nastavnik***. Pošto jedna beleška može biti vezana za jedan ili više kurseva, odnosno jednom objektu klase ***Beleska*** se pridružuje jedan ili više objekata klase ***BeleskaKurs***, u datoteci za preslikavanje definiše se da svakom objektu klase ***Beleska*** odgovara skup objekata klase ***BeleskaKurs***. Na isti način se definiše i odnos između klasa ***Beleska*** i ***BeleskaDosije***. Opisani su i atributi klase ***Beleska*** i koje im kolone odgovaraju u tabeli ***Beleska***.

Opis datoteke za preslikavanje tabele ***Kurs*** je takođe promenjen, dodat je opis da jednom objektu klase ***Kurs*** se pridružuje skup objekata klase

BeleskaKurs. Slična promena je izvršena i nad datotekom za preslikavanje tabele ***Dosije.***

Datoteka *BeleskaKurs.hbm.xml* koja opisuje preslikavanje tabele ***Beleska_Kurs*** u klasu ***BeleskaKurs*** izgleda:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="matf.studinfo.db.BeleskaKurs" table="BELESKA_KURS"
     schema="DBN">
        <composite-id name="id" class="matf.studinfo.db.BeleskaKursId">
            <key-property name="idBeleske" type="int">
                <column name="ID_BELESKE" />
            </key-property>
            <key-property name="idKursa" type="int">
                <column name="ID_KURSA" />
            </key-property>
        </composite-id>
        <many-to-one name="kurs" class="matf.studinfo.db.Kurs" update="false"
             insert="false" fetch="select">
            <column name="ID_KURSA" not-null="true" />
        </many-to-one>
        <many-to-one name="beleska" class="matf.studinfo.db.Beleska"
             update="false" insert="false" fetch="select">
            <column name="ID_BELESKE" not-null="true" />
        </many-to-one>
    </class>
</hibernate-mapping>
```

Slika 34: Datoteka za preslikavanje BeleskaKurs.hbm.xml

Pošto tabela ***Beleska_Kurs*** ima složen primarni ključ koji definišu kolone ***ID_BELESKE*** i ***ID_KURSA***, opisano je da jedan objekat klase ***BeleskaKursId*** predstavlja identifikator jednog objekta klase ***BeleskaKurs***. Definisani su i odnosi „više prema jedan“ između klasa ***BeleskaKurs*** i ***Beleska***, kao i između ***BeleskaKurs*** i ***Kurs***.

Nove konfigiracione datoteke moraju biti navedene u glavnoj konfiguracionoj datoteci da bi bile upotrebljene. Priložen je i deo datoteke *hibernate.cfg.xml*, definisane u okviru sistema StudInfo:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
       <property name="hibernate.connection.url">
             jdbc:db2://localhost:50000/studinfo
        </property>
       <property name="hibernate.connection.username">…</property>
       <property name="hibernate.connection.password">…</property>
```

```
                                    • • •
        <property name="hibernate.connection.driver_class">
              com.ibm.db2.jcc.DB2Driver
         </property>
        <property name="hibernate.dialect">
              org.hibernate.dialect.DB2Dialect
         </property>

                                    • • •
         <mapping resource="matf/studinfo/db/Nastavnik.hbm.xml" />
         <mapping resource="matf/studinfo/db/Fakultet.hbm.xml" />
         <mapping resource="matf/studinfo/db/Dosije.hbm.xml" />


                                    • • •

         <mapping resource="matf/studinfo/db/Beleska.hbm.xml" />
         <mapping resource="matf/studinfo/db/BeleskaKurs.hbm.xml" />
         <mapping resource="matf/studinfo/db/BeleskaDosije.hbm.xml" />

                                    • • •

    </session-factory>
</hibernate-configuration>
```

Slika 35: Konfiguracionoa datoteka hibernate.cfg.xml

4.4 JSP stranice

U sistemu StudInfo *JSP* stranice su podeljene na *JSP* stranice za prikazivanje i *JSP* akcione stranice. *JSP* stranice za prikazivanje proizvode izlaz koji se prikazuje korisniku i ne proizvode trajne promene u sistemu. Služe samo za prikaz podataka. *JSP* akcione stranice se sastoje samo od *JSP* elemenata i koriste za dodavanje novog ili menjanje postojećeg sadržaja u bazi podataka. One služe isključivo za izvođenje transakcija i ne sadrže nikakav kod za prikazivanje sadržaja.

Sve *JSP* stranice u sistemu StudInfo kao izlaz daju dokument u *XML* formatu koji odgovara internoj specifikaciji dokumenta u sistemu. *XML* dokument zatim prolazi kroz transformaciju i kao rezultat se dobija *HTML* stranica koja se šalje korisniku.

4.4.1 JSP stranice za prikazivanje

U JSP stranicama za prikazivanje, *JSP* elementi obrađuju zahtev korisnika i na osnovu njega određuju izgled stranice. Sastoje se iz dva dela: dela za pripremu i dela za pravljenje izveštaja.

Osnovni oblik *JSP* stranice za prikazivanje je prikazan na slici 36.

```
<%@ page language="java" extends="matf.studinfo.web.BaseJSPServlet"
   contentType="text/xml; charset=UTF-8" pageEncoding="UTF-8"%>
<%
   assertPermission(...);

   RequestParameters params = new RequestParameters(request);
   RequestParameters recovery = getRecoveryParams();
   ...
%>
<document>
   <head>
      <title>...</title>
   </head>
```

Slika 36: Osnovni oblik JSP stranice za prikazivanje

Prvo se navodi direktiva *page u* kojoj su opisana svojstava koja se odnose na *JSP* stranicu. Stranica nasleđuje servlet *BaseJSPServlet*, koji se koristi za sve *JSP* stranice za koje je neophodno da je korisnik prijavljen. Ako korisnik koji nije prijavljen pokuša da pristupi *JSP* stranici, automatski će se izvršiti preusmeravanje na početnu stranu aplikacije, na kojoj je formular za prijavljivanje.

U okviru *JSP* koda se najpre pomoću metoda *assertPermission* proverava da li prijavljeni korisnik ima pravo da pristupi stranici. Pomoću objekta klase *RequestParameters* pristupa se prosleđenim parametrima. Ako stranica sadrži formular mogu se proslediti podaci neophodni za oporavak od greške.

Osnovu napravljenog dokumenta predstavlja etiketa *document*, u okviru koje se definiše sadržaj stranice.

Kao primer *JSP* stranice za prikazivanje, u dodatku B je navedena *JSP* stranica *beleskaFormular.jsp*. Ona sadrži formular za dodavanje i ažuriranje nastavničke beleške čiji izgled zavisi od prosleđenih parametara.

4.4.2 JSP akcione stranice

JSP akcione stranice se sastoje samo od *JSP* elemenata. Krajni korisnik nije svestan postojanja akcionih stranica, jer ne proizvode nikakav izlaz za korisnika, već po obavljenom poslu preusmeravaju apikativni server na neku *JSP* stranicu za prikazivanje. *JSP a*akcione stranice koriste se za dodavanje novog ili menjanje postojećeg sadržaja u bazi podataka.

Osnovni oblik *JSP* akcione stranice je prikazan na slici 37.

```
<%@ page language="java" extends="matf.studinfo.web.ActionJSPServlet"
     contentType="text/xml; charset=UTF-8" pageEncoding="UTF-8"%>
<%
   assertPermission(...);
   RequestParameters params = new RequestParameters(request);

                          ...
```

```
    requestRedirectParam("JSPzaprikazivanje", "param", parameter);
    // ili samo requestRedirect("JSPzaprikazivanje ") ako nema parametara
%>
```

Slika 37: Osnovni oblik JSP stranice za prikazivanje

Prvo se navodi direktiva *page u* kojoj su opisana svojstva *JSP* akcione stranice. Stranica nasleđuje servlet *ActionJSPServlet* koji obezbeđuje da parametri i stranica sa koje je poslat formular budu zapamćeni. Ako dođe do greške, ponovo će biti prikazana stranica sa formularom koji je pokrenuo akcionu stranicu, a njegov sadržaj će biti popunjen vrednostima koje su tu bile pre slanja formulara. Servlet *ActionJSPServlet* nasleđuje servlet *BaseJSPServlet.*

U okviru *JSP* koda se najpre pomoću metoda *assertPermission* proverava da li prijavljeni korisnik ima pravo da pristupi stranici. Pomoću objekta klase *RequestParameters* pristupa se prosleđenim parametrima. Zatim se vrši dodavanje novog ili menjanje postojećeg sadržaja u bazi podataka, zavisno od parametara koji su prosleđeni.

Svaka *Home* klasa nasleđuje *GenericHibernateHome* klasu u kojoj su definisani metodi za čuvanje i brisnje objekata iz baze podataka (*saveNew* i *makeTransient*). Pravljenjem objekta neke od klasa *Home* implicitno se započinje transakcija [2] ili koristi već započeta ako postoji. Zato je prilikom brisanja ili čuvanja nekog objekta potrebno prvo napraviti objekat *Home* klase koja je pridružena klasi objekta koji se čuva ili briše iz baze podataka, i korišćenjem njegovih metoda izvršiti potrebnu operaciju.

Na kraju se vrši preusmeravanje na neku *JSP* stranicu za prikazivanje.

U dodatku B je priložena akciona stranica *beleska.jsp* koja se koristi za dodavanje nove beleške ili ažuriranje postojeće i sadrži preusmerenje na *JSP* stranicu za prikazivanje beleški *prikazBeleski.jsp*.

[2] Transakcija je skup operacija nad bazom podataka koji mora biti uspešno okončan pre nego što se rezultat u celosti upiše u bazu podataka.[13]

5 Zaključak

U radu je opisan podsistem za nastavničke beleške informacionog sistema StudInfo. Informacioni sistem StudInfo predstavlja veb aplikaciju koja omogućava redovan rad studentske službe, podršku nastavnog procesa i praćenje studentskih aktivnosti na više fakulteta Univerziteta u Beogradu.

Podsistem za nastavničke beleške je implementiran sa ciljem da se olakša i unapredi vođenje nastavnih aktivnosti vezanih za kurseve i studente. Time se smanjuje mogućnost zaboravljanja ili gubljenja važnih informacija koji se ne mogu drugačije zabeležiti u postojećem informacionom sistemu.

Podsistem za nastavničke beleške je omogućio nastavnicima postavljanje, menjanje, brisanje i pregledanje beleški uz kurseve i studente u čijoj nastavi učestvuju. Beleške mogu da se dele sa svim nastavnicima koji učestvuju u održavanju nastave na istim kursevima ili studentima uz koje su postavljene beleške. Takođe, beleške se mogu označiti kao podsetnici i takve beleške se prikazuju svaki put kada se nastavnik prijavi na sistem.

Podsistem je projektovan i implementiran primenom savremenih metoda za projektovanje i razvoj veb aplikacija. Za čuvanje podataka koristi se relacioni sistem za upravljanje bazom podataka *DB2 IBM*. Za dinamičko generisanje veb strana, čiji sadržaj zavisi od zahteva koji klijent šalje veb serveru korišćena je *Java Server Pages* (*JSP*) tehnologija. Za preslikavanje podataka između relacione baze podataka i klasa u objektno-orijentisanom jeziku *Java* korišćen je *Hibernate.*

Prilikom testiranja podsistema za nastavničke beleške uočeno je da postoji prostor za dalja unapređenja podsistema. Razdvajanjem datuma pravljenja i datuma poslednje izmene beleške, sačuvala bi se informacija postavljanja beleške, a prema datumu izmene bi mogao da se određuje prioritet pri prikazivanju beleški. Bilo bi dobro omogućiti da se jedna ista beleška odnosi i na studente i na kurseve jer je često potrebno zabeležiti podatak koji se odnose na studenta na nekom kursu. Ako bi i osoblje studentske službe moglo da vidi beleške koje su označene kao službene, ubrzala bi se komunikacija i sa studentskom službom. Ako postoji podsetnik uz kurs ili studenta, bilo bi dobro istići ga na stranicama posvećenim odgovarajućim kursevima i grupama ili studentima. Umesto podele na lične, službene ili lične i službene, možda bi bila bolja podela na „moje beleške“, „tuđe službene beleške“ i „sve beleške“. Osim unapređenja podsistema za nastvaničke beleške, ceo sistem bi mogao da se unapredi ako bi se omogućilo da i studenti vode sopstvene beleške uz kurseve koje slušaju.

Izradom master rada upoznala sam nove tehnologije u kojima prethodno nisam radila i uvidela da primenjena tehnologija u znatnoj meri olakšava razvijanje dinamičkih veb aplikacija. Svako ko ima dobro predzanje iz programskog jezika *Java*, *HTML*-a i *XML*-a, vrlo brzo može da nauči i

primeni *JSP* tehnologiju. Način na koji *Hiberante* obezbeđuje objektno-relaciono preslikavanje je lak za shvatiti i sa osnovnim znanjem objektno-orijentisanih jezika, relacionih baza podataka i *XML*-a, lako je razumeti datoteke za preslikavanje. Primenom alata *hbm2java* znatno se smanjuje vreme potrebno za razvoj veb aplikacije i smanjuje se mogućnost grešaka prilikom implementacije klasa podataka i datoteka za preslikavanje. Kako Internet ima sve veći značaj u savremenom društvu, što povlači i razvoj sve većeg broja veb aplikacija, sigurna sam da će mi ovo iskustvo koristiti u budućnosti.

6 Dodatak

6.1 Dodatak A – Klase podataka

6.1.1 Beleska

```
package matf.studinfo.db;

import java.util.Date;
import java.util.HashSet;
import java.util.Set;

public class Beleska implements java.io.Serializable {

   private int id;
   private Nastavnik nastavnik;
   private String tekst;
   private Date datum;
   private short podsetnik;
   private char tip;
   private Set<BeleskaDosije> beleskaDosijes = new HashSet<BeleskaDosije>(0);
   private Set<BeleskaKurs> beleskaKurses = new HashSet<BeleskaKurs>(0);

   public Beleska()
   {}

   public Beleska(int id, Nastavnik nastavnik, String tekst, Date datum,
                  short podsetnik, char tip)
   {
      this.id = id;
      this.nastavnik = nastavnik;
      this.tekst = tekst;
      this.datum = datum;
      this.podsetnik = podsetnik;
      this.tip = tip;
   }

   public Beleska(int id, Nastavnik nastavnik, String tekst, Date datum,
                  short podsetnik, char tip, Set<BeleskaDosije> beleskaDosijes,
                  Set<BeleskaKurs> beleskaKurses) {

      this.id = id;
      this.nastavnik = nastavnik;
      this.tekst = tekst;
      this.datum = datum;
      this.podsetnik = podsetnik;
      this.tip = tip;
      this.beleskaDosijes = beleskaDosijes;
      this.beleskaKurses = beleskaKurses;
   }

   public int getId() {
      return this.id;
   }
```

```
public void setId(int id) {
   this.id = id;
}

public Nastavnik getNastavnik() {
   return this.nastavnik;
}

public void setNastavnik(Nastavnik nastavnik) {
   this.nastavnik = nastavnik;
}

public String getTekst() {
   return this.tekst;
}

public void setTekst(String tekst) {
   this.tekst = tekst;
}

public Date getDatum() {
   return this.datum;
}

public void setDatum(Date datum) {
   this.datum = datum;
}

public short getPodsetnik() {
   return this.podsetnik;
}

public void setPodsetnik(short podsetnik) {
   this.podsetnik = podsetnik;
}

public char getTip() {
   return this.tip;
}

public void setTip(char tip) {
   this.tip = tip;
}

public Set<BeleskaDosije> getBeleskaDosijes() {
   return this.beleskaDosijes;
}

public void setBeleskaDosijes(Set<BeleskaDosije> beleskaDosijes) {
   this.beleskaDosijes = beleskaDosijes;
}

public Set<BeleskaKurs> getBeleskaKurses() {
   return this.beleskaKurses;
}

public void setBeleskaKurses(Set<BeleskaKurs> beleskaKurses) {
   this.beleskaKurses = beleskaKurses;
}
```

```
}
```

6.1.2 Klasa BeleskaKurs

```
package matf.studinfo.db;

public class BeleskaKurs implements java.io.Serializable {

   private BeleskaKursId id;
   private Kurs kurs;
   private Beleska beleska;

   public BeleskaKurs() {
   }

   public BeleskaKurs(BeleskaKursId id, Kurs kurs, Beleska beleska) {
      this.id = id;
      this.kurs = kurs;
      this.beleska = beleska;
   }

   public BeleskaKursId getId() {
      return this.id;
   }

   public void setId(BeleskaKursId id) {
      this.id = id;
   }

   public Kurs getKurs() {
      return this.kurs;
   }

   public void setKurs(Kurs kurs) {
      this.kurs = kurs;
   }

   public Beleska getBeleska() {
      return this.beleska;
   }

   public void setBeleska(Beleska beleska) {
      this.beleska = beleska;
   }

}
```

6.1.3 Klasa BeleskaKursId

```
package matf.studinfo.db;

public class BeleskaKursId implements java.io.Serializable {

   private int idBeleske;
   private int idKursa;
```

```
    public BeleskaKursId() {
    }

    public BeleskaKursId(int idBeleske, int idKursa) {
       this.idBeleske = idBeleske;
       this.idKursa = idKursa;
    }

    public int getIdBeleske() {
       return this.idBeleske;
    }

    public void setIdBeleske(int idBeleske) {
       this.idBeleske = idBeleske;
    }

    public int getIdKursa() {
       return this.idKursa;
    }

    public void setIdKursa(int idKursa) {
       this.idKursa = idKursa;
    }

    public boolean equals(Object other) {
       if ((this == other))
          return true;
       if ((other == null))
          return false;
       if (!(other instanceof BeleskaKursId))
          return false;
       BeleskaKursId castOther = (BeleskaKursId) other;

       return (this.getIdBeleske() == castOther.getIdBeleske())
            && (this.getIdKursa() == castOther.getIdKursa());
    }

    public int hashCode() {
       int result = 17;

       result = 37 * result + this.getIdBeleske();
       result = 37 * result + this.getIdKursa();
       return result;
    }

}
```

6.1.4 Klasa BeleskaHome

```
package matf.studinfo.db.home;

import matf.studinfo.db.*;
import matf.studinfo.db.utils.*;
import org.hibernate.*;
import java.util.*;
```

```
public class BeleskaHome extends GenericHibernateHome<Beleska, Integer>
{
   public BeleskaHome()
   {
      super(Beleska.class);
   }

   public static Comparator<Beleska> compareDate = new Comparator<Beleska>()
   {
      public int compare(Beleska b1, Beleska b2)
      {
         int res=-DataHelper.compareToIgnoreTime(b1.getDatum(), b2.getDatum());
         if(res==0)
            res= ((Integer)b1.getId()).compareTo((Integer)b2.getId());
         return(res);
      }
   };

   public List<Beleska> beleskeZaNastavnika(NastavnikId nastavnikId, Integer
                                            kursId, Integer indeks)
   {
      String queryStr = "" +
      "SELECT  {b.*} ";

      if(kursId != null)
         queryStr += "" +
               " FROM dbn.beleska b JOIN dbn.beleska_kurs bk" +
               " ON b.id=bk.id_beleske" +
               " WHERE (b.id_nastavnika= :nId AND " +
                  " b.id_fakulteta= :fId AND bk.id_kursa= :kId)" +
               " OR (b.id_nastavnika != :nId AND " +
                  " b.id_fakulteta = :fId AND bk.id_kursa= :kId )" +
               " ORDER BY b.datum desc";

      else if(indeks!=null)
         queryStr +="" +
               " FROM dbn.beleska b JOIN dbn.beleska_dosije bd" +
               " ON b.id=bd.id_beleske AND b.id_fakulteta=bd.id_fakulteta" +
               " WHERE (b.id_nastavnika= :nId AND b.id_fakulteta= :fId AND " +
                           " bd.indeks= :ind) " +
               " OR (b.id_nastavnika != :nId AND b.id_fakulteta= :fId AND " +
                           " bd.indeks= :ind) " +
               " ORDER BY b.datum desc";
      else
         queryStr +="" +
               " FROM dbn.beleska b " +
            " WHERE (b.id_nastavnika= :nId AND b.id_fakulteta= :fId) " +
            " OR ( b.id_nastavnika != :nId AND b.id_fakulteta= :fId )" +
            " ORDER BY b.datum desc";

   Query query = HibernateUtil.getSession()
                .createSQLQuery(queryStr)
                   .addEntity("b", getPersistentClass())
                   .setInteger("nId", nastavnikId.getIdg())
                   .setInteger("fId", nastavnikId.getIdFakulteta());

   if(kursId != null)
      query.setInteger("kId", kursId);
   else if(indeks != null)
```

```
   query.setInteger("ind", indeks);

List<Beleska> list = query.list();

return(list);
}

public Integer getBaseId()
{
   Number base = (Number)getSession()
      .createQuery("SELECT COALESCE(MAX(id), 0) FROM Beleska")
      .uniqueResult();

   return(base.intValue());
}

public Integer getNextId(Integer prevId)
{
   return(prevId + 1);
}

public boolean postojiBeleskaPodsetnik(NastavnikId nastavnikId)
{
   String queryStr = "" +
   "SELECT  {b.*} " +
   " FROM dbn.beleska b " +
   " WHERE b.id_nastavnika= :nId AND b.id_fakulteta= :fId AND " +
         " b.podsetnik = 1 ";

Query query = HibernateUtil.getSession()
             .createSQLQuery(queryStr)
                .addEntity("b", getPersistentClass())
                .setInteger("nId", nastavnikId.getIdg())
                .setInteger("fId", nastavnikId.getIdFakulteta());

List<Beleska> list = query.list();

if(list.isEmpty())
   return false;

return true;
}

public List<Beleska> beleskePodsetnik(NastavnikId nastavnikId,
                                      boolean sort)
{
   String queryStr = "" +
   "SELECT  {b.*} " +
   " FROM dbn.beleska b " +
   " WHERE b.id_nastavnika= :nId AND b.id_fakulteta= :fId AND " +
           " b.podsetnik = 1 ";

Query query = HibernateUtil.getSession()
             .createSQLQuery(queryStr)
                .addEntity("b", getPersistentClass())
                .setInteger("nId", nastavnikId.getIdg())
                .setInteger("fId", nastavnikId.getIdFakulteta());

List<Beleska> list = query.list();
```

```
    if(sort)
         Collections.sort(list, BeleskaHome.compareDate);

    return (list);
    }
}
```

6.2 Dodatak B – JSP stranice

6.2.1 JSP stranice za prikazivanje

Kao primer *JSP* stranice za prikazivanje priložena je *JSP* stranica *beleskaFormular.jsp*. Ona sadrži formular za dodavanje i ažuriranje nastavničke beleške, čiji izgled zavisi od prosleđenih parametara.

Pored neophodnih opisanih *JSP* elemenata koriste se i dodatni *JSP* elementi. Prvo su navedene direktive *page*, kojima su opisana svojstava koja se odnose na *JSP* stranicu, i direktiva *include*, kojom je definisano da se umeće sadržaj *JSP* stranice *beleska.jspf.* U segmentu *JSP* stranice *beleska.jspf* su definisane pomoćne metode koje se koriste u nekoliko *JSP* stranica koje rade sa beleškama.

U samoj *JSP* stranici, u okviru deklaracije, definisane su neophodne pomoćne metode, a u obliku skriptleta je definisan izgled stranice, koji zavisi od prosleđenih parametara.

U skriptletu za prikaz podataka koriste se objekti klasa komponenata poput *CheckBox* ili *ComboBox,* koje je razvio razvojni tim informacionog sistema StudInfo. U okviru elementa *document* se pored statičkog sadržaja koriste i *JSP* elementi (izrazi) za pristupanje objektima klasa programskog jezika *Java*, definisanim u skriptletu (formi *form*), i za kreiranje novih objekata (klase *ParamLink*). Definisane su i *JavaScript* funkcije koje obezbeđuju proveru forme i neke funkcionalnosti korisničkog interfejsa na klijentskoj strani.

```
<%@ page language="java" extends="matf.studinfo.web.BaseJSPServlet"
   contentType="text/xml; charset=UTF-8"
     pageEncoding="UTF-8"%>
<%@ page import="matf.studinfo.db.*" %>
<%@ page import="matf.studinfo.db.home.*" %>
<%@ page import="matf.studinfo.web.*" %>
<%@ page import="matf.studinfo.web.components.*" %>
<%@ page import="java.util.List" %>
<%@ page import="matf.studinfo.exceptions.AppPermissionException"%>
<%@ include file="beleska.jspf" %>
<%!

public static void dodavanjeNovihStudenata(RequestParameters params,
   List<String> izborNastavnika, Table studentiZaBelesku, Form form,
   Nastavnik nastavnik )
```

```
{
   DosijeHome dh = new DosijeHome();
   Set<Dosije> dodatiStudenti = new HashSet<Dosije>(dh
         .fromEncodedIds(params.getValues("noviStudenti"),
            "Један од студената не постоји у бази"));

   List<Dosije> sviStudenti = new ArrayList<Dosije>(dodatiStudenti);
   String[] postojeciStudentiString = params.getValues("studenti");

   for(String s : postojeciStudentiString)
   {
      Dosije d = new DosijeHome()
                     .findById(new DosijeId(nastavnik.getId().getIdFakulteta(),
                                    new Integer(s)));
      sviStudenti.add(d);
   }


   Collections.sort(sviStudenti, DosijeHome.surnameAndNameComparator);

   for(int i=sviStudenti.size()-1; i>0; i--)
         if(sviStudenti.get(i).equals(sviStudenti.get(i-1)))
            sviStudenti.remove(i);

   CheckBox chBox;
   for(Dosije ps: sviStudenti)
   {
      boolean izabran=false;
      if (CheckBox.booleanFromParameters(params, "d-" + ps.getId().getIndeks())
          || dodatiStudenti.contains(ps))
         izabran=true;

      chBox = new CheckBox("d-" + ps.getId().getIndeks(), izabran);
      chBox.setStyle("vertical-align: middle;");

      studentiZaBelesku.addRow()
         .addData(new LabelContainer()
         .add(chBox)
         .add(new Item(ps.getImePrezimeIndeks())));

      izborNastavnika.add(chBox.getName());
      form.addHiddenParameter("studenti", ps.getId().getIndeks().toString());
   }
   }

public static void prikazTekucihKurseva(List<Kurs> listaKursevaNastavnika,
   Table sviKurseviNastavnika, List<String> izborNastavnika, Kurs kurs,
   Beleska beleska, short godina)
{
   if(!listaKursevaNastavnika.isEmpty())
   {
      sviKurseviNastavnika.addRow()
         .addData(new TableData("Курсеви за које сам координатор у " + godina +
                                    "/"+ (godina+1) +". школској години")
                  .addAttribute("style", "font-weight: bold; font-size: 14px;"));

      sviKurseviNastavnika.addRowSeparator(2, 8);

      for (Kurs k : listaKursevaNastavnika)
```

```
      {
         boolean selectedKurs= kurs != null && k.getId().equals(kurs.getId());

         if(beleska != null)
            for( BeleskaKurs bk : beleska.getBeleskaKurses())
               if(bk.getKurs().getId().equals(k.getId()))
                     selectedKurs=true;

         CheckBox chBox = new CheckBox("k-" + k.getId(), selectedKurs);
         chBox.setStyle("vertical-align: middle;");

         sviKurseviNastavnika.addRow()
            .addData(new LabelContainer()
                  .add(chBox)
                  .add(new Item(k.getNazivISifra())));

         izborNastavnika.add(chBox.getName());
      }
   }
}
%>
<%
   assertTeacher();

   RequestParameters params = new RequestParameters(request);
   RequestParameters recovery = getRecoveryParams();

   KursHome kh = new KursHome();
   Kurs kurs=null;
   Beleska beleska = null;
   Dosije dosije= null;
   DosijeHome dh = new DosijeHome();

   short godina=new FakultetHome().getCurrentSchoolYear();
   Nastavnik nastavnik = getSessionData().getTeacherFromDB();
   NastavnikId nId = nastavnik.getId();
   Integer fId = getSessionData().getFacultyConfig().getFacultyId();

   Form form = new Form("beleska", "beleska", recovery);
   Form form2 = new Form("beleskaFormular", "beleskaFormular", recovery);

   ParamLink prikazBeleski;

   Table sviKurseviNastavnika = new Table("kurseviNastavnika");
   Table studentiZaBelesku = new Table("studentiZaBelesku");
   List<Kurs> listaKursevaNastavnika = kurseviNastavnikUTekucojGodini(nId,
                                          godina, true);
   List<String> izborNastavnika = new ArrayList<String>(0);


   if (params.get("kurs") != null)
   {
      kurs = new KursHome().fromHttpRequest(params, "kurs",
            "Није наведен идентификатор курса",
            "Одабрани курс не постоји у бази");

      form.addHiddenId("kurs", kurs);
      prikazBeleski = new ParamLink("Белешке за курс " +
                        kurs.getPredmetPlanaStudija().getPredmet()
```

```
                        .getNazivISifra(), "prikazBeleski", "kurs", kurs);
}
else if (params.get("dosije") != null)
{
   dosije = new DosijeHome().fromHttpRequest(params, "dosije",
        "Није наведен идентификатор студента",
        "Одабрани студент не постоји у бази");

   form.addHiddenParameter("studenti",
                           dosije.getId().getIndeks().toString());
   prikazBeleski = new ParamLink("Белешке за студента " +
                                    dosije.getImePrezimeIndeks(),
                                 "prikazBeleski", "dosije", dosije);
}
else if(params.get("beleska") != null)
{
     beleska = new BeleskaHome().fromHttpRequest(params, "beleska",
          "Није наведен идентификатор белешке",
          "Одабрана белешка не постоји у бази");

     form.addHiddenId("beleska", beleska);
     prikazBeleski = new ParamLink("Приказ белешки ", "prikazBeleski");
}
else
{
   prikazBeleski = new ParamLink("Приказ белешки ", "prikazBeleski");
}

Table tekstIPodsetnikBeleske = new
Table("tekstBeleske").addAttribute("width", "60%")
.setStyle("border-style: solid; border-width: 1px; border-color: #D5D8E6;");

TextArea tekst = new TextArea("tekst")
.addMaxLengthRestriction("Тест белеске", Beleska.class, "tekst")
.setStyle("width: 550px; height: 200px;");

CheckBox podsetnikBeleska = new CheckBox("podsetnik");
ComboBox tipCombo = new ComboBox("tip");

tipCombo.addItem("Лична", "l");
tipCombo.addItem("Службена", "s");

if(params.get("tekst") != null)
   tekst.setText(params.get("tekst"));
else if (beleska != null)
   tekst.setText(beleska.getTekst());

if(params.get("podsetnik") != null)
{
   if(CheckBox.booleanFromParameters(params, "podsetnik"))
      podsetnikBeleska.setChecked(true);
}
else if(beleska != null)
   if(beleska.getPodsetnik()== 1)
      podsetnikBeleska.setChecked(true);

if(params.get("tip") != null)
{
   if(params.get("tip").equals("s"))
```

```
      tipCombo.selectItemByValue("s");
}
else if(beleska != null)
   if(beleska.getTip()== 's')
      tipCombo.selectItemByValue("s");

tekstIPodsetnikBeleske.setCaption("Белешка");
tekstIPodsetnikBeleske.addRow()
                        .addHeader("Текст").addData(tekst);
tekstIPodsetnikBeleske.addRow()
                        .addHeader("Подсетник").addData(podsetnikBeleska);
tekstIPodsetnikBeleske.addRow()
                        .addHeader("Тип")
                        .addData(tipCombo);

if(beleska != null)
{
   CheckBox datumBeleske = new CheckBox("datum");
   tekstIPodsetnikBeleske.addRow()
            .addHeader("Ажурирај датум на данашњи").addData(datumBeleske);
}


if(kurs != null || ( beleska != null &&
                        !beleska.getBeleskaKurses().isEmpty()))
{

   prikazTekucihKurseva(listaKursevaNastavnika, sviKurseviNastavnika,
                           izborNastavnika,kurs, beleska, godina);

}
else if (beleska != null && !beleska.getBeleskaDosijes().isEmpty())
{
   studentiZaBelesku.addRow()
      .addData(new TableData("Студенти")
         .addAttribute("style", "font-weight: bold; font-size: 14px;"));
   studentiZaBelesku.addRowSeparator(2, 8);

   if(params.get("noviStudenti") != null)
   {
      dodavanjeNovihStudenata(params, izborNastavnika, studentiZaBelesku,
                              form, nastavnik);
   }
   else
   {
      for(BeleskaDosije bd : beleska.getBeleskaDosijes())
      {

         CheckBox chBox = new CheckBox("d-" +
                           bd.getDosije().getId().getIndeks(), true);
         chBox.setStyle("vertical-align: middle;");

         studentiZaBelesku.addRow()
            .addData(new LabelContainer()
               .add(chBox)
               .add(new Item(bd.getDosije().getImePrezimeIndeks())));

         form.addHiddenParameter("studenti",
```

```
        bd.getDosije().getId().getIndeks().toString());

        izborNastavnika.add(chBox.getName());
      }
   }
}
else if (dosije != null)
{
   studentiZaBelesku.addRow()
      .addData(new TableData("Студенти")
        .addAttribute("style", "font-weight: bold; font-size: 14px;"));
   studentiZaBelesku.addRowSeparator(2, 8);

   CheckBox chBox = new CheckBox("d-" + dosije.getId().getIndeks(), true);
   chBox.setStyle("vertical-align: middle;");

   studentiZaBelesku.addRow()
      .addData(new LabelContainer()
        .add(chBox)
        .add(new Item(dosije.getImePrezimeIndeks())));

   izborNastavnika.add(chBox.getName());

}
else if(params.get("noviStudenti") != null)
{
   dodavanjeNovihStudenata(params, izborNastavnika, studentiZaBelesku, form,
                                nastavnik);
}
else if (!listaKursevaNastavnika.isEmpty())
{
   prikazTekucihKurseva(listaKursevaNastavnika, sviKurseviNastavnika,
                            izborNastavnika,kurs, beleska, godina);
}
else
   form.addElement(
        new Span()
             .add("У "+ godina + "/"+ (godina+1) +". школској години нема
                    курсева за које можете да поставите белешку."));

if (izborNastavnika.size() > 1)
{
   if(params.get("noviStudenti") != null)
      studentiZaBelesku.addRow()
      .addData(new TableData(new ParamLink("Означи све студенте",
                          "javascript: ;")
                          .addAttribute("onClick",
                          "oznaciCheckboxove("
                          + Element.createJSArray(izborNastavnika, true)
                          + ", true);")));
   else
      sviKurseviNastavnika.addRow()
      .addData(new TableData(new ParamLink("Означи све курсеве",
                 "javascript: ;")
             .addAttribute("onClick",
                  "oznaciCheckboxove("
                    + Element.createJSArray(izborNastavnika, true)
                    + ", true);")));
```

```
   }


   if(dosije != null || params.get("noviStudenti") != null || (beleska != null
        && !beleska.getBeleskaDosijes().isEmpty()))
      form2
      .addElement(new Div()
           .addStyle("text-align: right;")
           .add(new EditLink("Додај студенте за белешку",
                                "izborStudenataZaBelesku")
              .setSize(640, 480)
              .addParam("jsFunkcija", "dodajStudenta")));


      Element<?> submit = new InputSubmit(beleska != null ? "Промени белешку" :
                           "Постави белешку")
                                .setClassAttribute("non-printable");

      submit.addAttribute("onClick", "return(validateAndSubmit(this.form));");

      Element<?> reset = new InputReset("Поништи измене")
                                .setClassAttribute("non-printable");

      form.setSubmitReset(new SubmitReset(submit, reset));

      form.addElement(tekstIPodsetnikBeleske)
         .addBreak()
         .addBreak();

      if(dosije != null || params.get("noviStudenti") != null ||
         (beleska != null && !beleska.getBeleskaDosijes().isEmpty()))
         form.addElement(studentiZaBelesku);
      else
         form.addElement(sviKurseviNastavnika);

%>
<document>
   <head>
      <title><%= beleska == null
         ? "Нова белешка"
         : ("Белешка додата "
           + DataHelper.formatDate(beleska.getDatum(), DataHelper.CASE_GENITIV)
        + " у " + DataHelper.formatTimeOfDay(beleska.getDatum(), false, false))
   %></title>
   </head>
   <body>
   <script language="javascript">
           function oznaciCheckboxove(kursevi)
           {
              var form = document.forms['<%= form.getName() %>'];

              for(var i = 0; i &lt; kursevi.length; i++)
                 form[kursevi[i]].checked = true;
           }

           function postojiStiklirano(kursevi)
           {
              var form = document.forms['<%= form.getName() %>'];
```

```
      for(var i = 0; i &lt; kursevi.length; i++)
         if( form[kursevi[i]].checked)
               return true;
      return false;
   }

   function validateAndSubmit(form)
   {
      var invalidFields = new Array();
      var valid = true;

      var kursevi = <%=
          Element.encode(Element.createJSArray(izborNastavnika, true),
                         false) %>;
      var posStik =postojiStiklirano(kursevi);

   if (!posStik)
   {

      if (<%= beleska == null %>)
      {
         addFormValidationErrorString(invalidFields, form,
               new ValField(kursevi, '', 'Курсеви'),
               'Нису одабрани курсеви на које се односи белешка');
      }
      else
      {
         if (!confirm('Нисте обележили ниједан курс , '
            + 'па ће белешка бити обрисана.\n\n'
            + 'Да ли заиста желите да обришете белешку?'))
            valid = false;
      }
   }
   else
   {
      var tekstEl = form['<%= tekst.getName() %>'];

      if (trim(tekstEl.value) == '')
         addFormValidationErrorString(invalidFields, form,
        new ValField(tekstEl.name, '', 'Текст белешке'),
                    'Мора бити унет текст');
   }

   if (invalidFields.length &gt; 0)
   {
      reportEditFormValidation(invalidFields);
      valid = false;
   }

   if (valid)
      valid = submitForm(form);

   return(valid);
}

function dodajStudenta(spisak)
{
   if (spisak.length &gt; 0)
   {
```

```
            var form = document.forms['<%= form.getName() %>'];

            for (var i = 0; i &lt; spisak.length; i++)
            {
                 var el = document.createElement("input");

                 el.setAttribute("type", "hidden");
                 el.setAttribute("name", "noviStudenti");
                 el.setAttribute("value",spisak[i].kodiraniID);
                 form.appendChild(el);
            }

            form.action="beleskaFormular";
            var dugme = form['<%= submit.getName() %>'];

            var valid = submitForm(form);
            dugme.disabled = false;

            return(valid);
          }
        }

        </script>
      <%= form %>
      <%= form2 %>

        <quick-menu>
        <item><%= prikazBeleski%></item>
        <item><%= new ParamLink("Курсеви за које сам координатор", "kursevi")
  %></item>
        <item><%= new ParamLink("Моје групе", "nastava") %></item>
     </quick-menu>

  </body>
</document>
```

6.2.2 JSP akcione stranice

Kao primer *JSP* akcione stranice priložena je *JSP* stranica *beleska.jsp* koja se koristi za dodavanje nove beleške ili ažuriranje postojeće i sadrži preusmerenje na *JSP* stranicu za prikazivanje beleški *prikazBeleski.jsp*.

```
<%@page import="com.sun.corba.se.spi.protocol.RequestDispatcherDefault"%><%@
  page language="java" extends="matf.studinfo.web.ActionJSPServlet"
  contentType="text/xml; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="matf.studinfo.db.*" %>
<%@ page import="matf.studinfo.db.home.*" %>
<%@ page import="matf.studinfo.exceptions.*" %>
<%@ page import="matf.studinfo.web.*" %>
<%@ page import="matf.studinfo.web.components.*" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.Date" %>
<%@ include file="beleska.jspf" %>
<%
  assertTeacher();

  RequestParameters params = new RequestParameters(request);
```

```
BeleskaHome bh = new BeleskaHome();
BeleskaKursHome bkh = new BeleskaKursHome();
BeleskaDosijeHome bdh = new BeleskaDosijeHome();

short godina=new FakultetHome().getCurrentSchoolYear();
String tekst = params.getRequired("tekst",
                                    "Није прослеђен текст обавештења.");
char tip = params.getRequired("tip", "l").charAt(0);

Nastavnik nastavnik = getSessionData().getTeacherFromDB();
NastavnikId nId = nastavnik.getId();
Date vreme = new Date();

Kurs kurs=null;
Beleska postojecaBeleska=null;
String[] studenti = null;

if (params.get("kurs") != null)
{
   kurs = new KursHome().fromHttpRequest(params, "kurs",
         "Није наведен идентификатор курса",
         "Одабрани курс не постоји у бази");
}
else if (params.get("beleska") != null)
{
   postojecaBeleska = new BeleskaHome().fromHttpRequest(params, "beleska",
         "Није наведен идентификатор белешке",
         "Одабрана белешка не постоји у бази");
}
else
   postojecaBeleska = null;

if(params.get("studenti") != null)
   studenti= params.getValues("studenti");

if(postojecaBeleska != null)
{
   Set<BeleskaKurs> kurseviBeleske=postojecaBeleska.getBeleskaKurses();
   Set<BeleskaDosije> studentiBeleske=postojecaBeleska.getBeleskaDosijes();

   if(!kurseviBeleske.isEmpty())
      for(BeleskaKurs bk : kurseviBeleske)
         bkh.makeTransient(bk);

   if(!studentiBeleske.isEmpty())
      for(BeleskaDosije bd : studentiBeleske)
         bdh.makeTransient(bd);

   if (!CheckBox.booleanFromParameters(params, "datum"))
      vreme=postojecaBeleska.getDatum();

   bh.makeTransient(postojecaBeleska);
}

Integer nextId=bh.getBaseId();

Beleska novaBeleska = new Beleska();
novaBeleska.setNastavnik(nastavnik);
```

```
novaBeleska.setTekst(tekst);
novaBeleska.setDatum(vreme);
novaBeleska.setTip(tip);


if (CheckBox.booleanFromParameters(params, "podsetnik"))
   novaBeleska.setPodsetnik((short)1);
else
   novaBeleska.setPodsetnik((short)0);

novaBeleska.setId(bh.getNextId(nextId));

List<BeleskaKurs> noveBeleskeKurs= null;
List<BeleskaDosije> noveBeleskeDosije = null;

if(studenti == null)
{
   noveBeleskeKurs = new ArrayList<BeleskaKurs>();
   Set<Integer> kurseviSaBeleskama = new HashSet<Integer>();

   List<Kurs> listaKursevaNastavnika = kurseviNastavnikUTekucojGodini(nId,
                                          godina, true);

   for(Kurs k : listaKursevaNastavnika)
   {
      if (CheckBox.booleanFromParameters(params, "k-" + k.getId()))
      {
         BeleskaKurs beleskaKurs = new BeleskaKurs();
         BeleskaKursId beleskaKursId = new BeleskaKursId(novaBeleska.getId(),
                                                         k.getId());

         beleskaKurs.setId(beleskaKursId);
         noveBeleskeKurs.add(beleskaKurs);
      }
   }
}
else
{
   noveBeleskeDosije = new ArrayList<BeleskaDosije>();
   Set<Integer> studentiSaBeleskama = new HashSet<Integer>();

   List<Dosije> listaStudenataNastavnika = studentiNastavnika(nId, godina);

   Integer maxRbr;

   for (String s : studenti)
   {
        Integer indeks= new Integer(s);

      if (CheckBox.booleanFromParameters(params, "d-" + s))
      {
         DosijeId studentId= new DosijeId(nastavnik.getId().getIdFakulteta(),
                                          indeks);
         BeleskaDosije beleskaDosije = new BeleskaDosije();
         BeleskaDosijeId beleskaDosijeId =
         new BeleskaDosijeId(novaBeleska.getId(), studentId.getIdFakulteta(),
                             studentId.getIndeks());

         beleskaDosije.setId(beleskaDosijeId);
```

```
            noveBeleskeDosije.add(beleskaDosije);

         }
      }
   }

   if(studenti != null)
   {
      if (!noveBeleskeDosije.isEmpty())
      {
         bh.saveNew(novaBeleska);

         for(BeleskaDosije ok : noveBeleskeDosije)
            bdh.saveNew(ok);
      }
   }
   else
   {
      if (!noveBeleskeKurs.isEmpty())
      {
         bh.saveNew(novaBeleska);

         for(BeleskaKurs ok : noveBeleskeKurs)
            bkh.saveNew(ok);
      }
   }

      requestRedirectParam("prikazBeleski", "beleska", novaBeleska);

%>
```

7 Reference

[1] *Business Process Visual ARCHITECT* (http://www.visual-paradigm.com/product/bpva/)

[2] *Visual Paradigm for UML* (http://www.visual-paradigm.com/product/vpuml/)

[3] *Database Visual ARCHITECT* (http://www.visual-paradigm.com/product/dbva/)

[4] *Eclipse* (http://www.eclipse.org/)

[5] Gordana-Pavlović Lažetić, „Osnove relacionih baza podataka“, drugo izdanje, 1999 (http://poincare.matf.bg.ac.rs/~gordana//projektovanjeBP/ER.pdf)

[6] Apache Tomcat, Apache Software Foundation (http://tomcat.apache.org/)

[7] Apache Tomcat – History (http://www.experiencefestival.com/a/Apache_Tomcat_-_History/id/4798959)

[8] Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock, „The J2EE™ 1.4 Tutorial” (http://download.oracle.com/docs/cd/E17802_01/j2ee/j2ee/1.4/docs/tutorial-update2/doc/J2EETutorial.pdf)

[9] Giulio Zambon , Michael Sekler „Beginning JSP, JSF, and Tomcat Web Development: From Novice to Professional”, 2007

[10] Hibernate (http://www.hibernate.org/)

[11] Christian Bauer, Gavin King, „Java persistence with Hibernate”, 2007

[12] Dijagrami toka podataka (http://www.vps.ns.ac.rs/nastavnici/Materijal/mat572.pdf)

[13] Saša Malkov, „Vežbe iz Baze podataka”, skripta, 1999.