Web aplikacija za profesore i studente

Maja Vasić*, Irena Serna*, Belma Ramić *, Dženan Riđanović** i Vensada Okanović*
* Elektrotehnički fakultet Univerziteta u Sarajevu, Sarajevo, BiH
** Université Lavala, Québec, Canada
t95lli@yahoo.com, irena_033@yahoo.com, belma_ramic@yahoo.com,
dzenan.ridjanovic@fsa.ulaval.ca, vensada.okanovic@etf.unsa.ba

***Apstrakt*—Na četvrtoj godini Elektrotehničkog fakulteta, Odsjek za računarstvo i informatiku, u okviru predmeta "Projektovanje sistemskog softvera" implementirana je web aplikacija za profesore i studente. Ova web aplikacija je napravljena korištenjem mvcLite framework-a, kojeg su studenti tokom semestra spiralno razumijevali. Web aplikacija za studente i profesore se sastoji od *profesorskog dijela* i *studentskog dijela* na koje se korisnik usmjerava u zavisnosti od korisničke prijave.**

Na početku kursa znanje studenata u radu s Java programskim jezikom, web aplikacijama i framework-om je bilo minimalno. Tako je razvoj web aplikacije izgledao kao nemoguća misija. Ali, korištenjem ranije razvijenog mvcLite framework-a, gdje je 80% generičkog kôda, za vrlo kratko vrijeme uspješno je implementirana kvalitetna web aplikacija.

***Ključne riječi:* web frameworks, dinamičke web aplikacije, learning process.**

I. UVOD

U današnje vrijeme upotreba računara zakoračila je u različite aspekte života. Tako je i sa obrazovnim institucijama, gdje se nastoji modernizirati nastavni proces i olakšati profesorima i studentima da zajednički dođu do cilja, znanja. Današnja ekspanzija Interneta i web aplikacija jedno je od takvih mogućih rješenja. U tu svrhu nastala je i ova web aplikacija, kao rezultat rada na četvrtoj godini Elektrotehničkog fakulteta u okviru predmeta "Projektovanje sistemskog softvera".

Cilj date aplikacije je prikazati moguće rješenje i implementaciju web aplikacija u nastavnom procesu. Pomoću nje studenti se informišu, prikupljaju materijale i informacije i na kraju polažu ispite. Ova aplikacija olakšava praćenje procesa polaganja ispita i uvid u postignute rezultate. S druge strane, profesor na jednostavniji način komunicira i ostvaruje saradnju sa studentima.

Ono što nećemo pokušati ovim radom, jeste upoznati čitaoca sa Javom, ili bilo kojim drugim programskim jezikom, kao ni radom sa bilo kojim od alata. Želimo samo predstaviti pristup izgadnje jedne web aplikcije; jedan specifičan pristup koji je nama omogućio da kroz relativno kratak vremenski period izgradimo funkcionalnu web aplikaciju.

II. PRISTUP

Ova web aplikacija je realizovana koristeći se spiralnim pristupom razvoju softvera. Prva spirala ili verzija aplikacije je koristila mali dio modela klasa sa samo nekoliko osnovnih funkcija. Sa svakom novom spiralom, novi dio modela je uveden i nove funkcije su dodane. To nam je omogućilo da razvijemo sigurnost u naše sposobnosti. Razlika između prve i zadnje spirale je ogromna.

Razvoj dinamičkih web aplikacija je kompleksan i zahtjevan proces. To podrazumjeva znanje u različitim oblastima, kao što su framework, Java Server Pages (JSP), Servlet, Tomcat i druge, zatim dizajn i programiranje (HTML, XML, Java, ...), korištenje različitih alata koji pomažu pri kreiranju web stranica poput Dreamweaver-a, Amaya-e, Photoshop-a, Corel-a, Flash-a, te mnoštva drugih pogodnih alata.

Danas postoji veliki broj načina kako napraviti kvalitetnu i funkcionalnu aplikaciju koja bi trebala poslužiti svojoj svrsi, tj. predstaviti pojedinca ili skupinu na Internetu. Međutim, koji od tih načina je najpogodniji, kako napraviti brzo i efikasno aplikaciju, a pri tome ne izmišljati «toplu vodu»? Čemu graditi nešto iznova, ispočetka, ako već postoji okvir ili ljuska po kojoj se istorodne aplikacije mogu generisati sa malom promjenom koda? Takva ljuska na osnovu koje nastaju aplikacije naziva se framework.

Frameworks postaju sve popularniji u softver inžinjeringu. Razlog za njihovu popularnost je povećan nivo produktivnost razvoja softvera. Framework je kolekcija interfejsa, apstraktnih klasa i realizovanih klasa koje nude generalna rješenja za određeni tip softvera. U slučaju web aplikacija, većina framework-a koriste MVC (Model View Controller) pattern za arhitekturu softvera.

Model se koristi za organizaciju podataka u vidu modela klasa i omogućava persistentnost podataka u bazama podataka ili datotekama. Pogledi (Views) su prezentacija podataka u vidu ekrana ili stranica koji omogućavaju korisnicima da rade ono što je dizajner softvera i naumio. Kontrolor (Controller) delegira zahtjeve korisnika kontrolnim akcijama koje diriguju komunikaciju između modela podataka i korisničkih pogleda.

MvcLite je framework baziran na MVC pattern-u. Cilj framework-a je pedagoški, da omogući studentima da relativno brzo razviju web aplikaciju. Zahvaljujući mvcLite framework-u uspijeli smo da razvijemo našu

aplikaciju u rekordnom vremenu u zadnjoj trećini kursa. Prve dvije trećine kursa su nam omogućile da upoznamo mvcLite framewok spiralno. Prvih nekoliko sedmica su bile uvodne, a onda je svaka nova sedmica donosila novu funkcionalnost framework-a. Tako smo prvo razumjeli šta je to framework i kakvu produktivnost nudi, a onda smo počeli da upoznajemo M, V i C komponente framework-a.

Naše znanje o Javi nije bilo veliko prije kursa. Takođe nismo dobro poznavali Eclipse (IDE za programiranje u Javi), Tomcat (web server), JSP i Servlet tehologije Jave za servere. Naše znanje koncepata framework-a i pattern-a nije bilo na zavidnom nivou. Kada se sve to uzme u obzir, lako je shvatiti da nismo vjerovali da ćemo biti u stanju razviti kompletnu web aplikaciju do kraja kursa. Ipak, aplikacija u svojoj finalnoj verziji sadrži 46 klasa, 40 .jsp stranica, te oko 19.000 linija koda.

Dodatno, ne samo da smo koristili mvcLite kroz Javine interfejse, nego smo i dobili osnovno obrazovanje iz softver inžinjeringa. Sada nam je jasno kakva je arhitektura framework-a i zašto su oni trenutno najproduktivnije sredstvo za razvijanje različitih tipova softvera, a naročito u bazama podataka i web sistemima. Sa poznavanjem ovog framework-a sigurni smo da će se biti laške u budućnosti uhvatiti u koštac sa tri najpopularnija framework-a: Hibernate za rad sa bazama podataka, Struts za web aplikacije i Spring za kontrolu softvera.

III. Model-View-Controller

A. *Osnovni koncepti*

Model-View-Controller arhitektura danas je široko rasprostranjen pristup za razvoj dinamičkih aplikacija. Ovaj koncept dijeli funkcionalnosti među objektima koji predstavljaju i čuvaju podatke, te na taj način smanjuje stepen zavisnosti među tim objektima. Tradicionalne dijelove zadatka kao što su ulaz, procesiranje i izlaz MVC mapira u korisničko interaktivno okruženje.

MVC predstavlja pristup kod kojeg su korisnički zahtjev, procesiranje zahtjeva i grafički odziv na taj zahtjev, eksplicitno razdvojeni i podržani sa tri tipa objekata, pri čemu je svaki od njih specijaliziran za određenu vrstu zadataka.

MVC arhitektura dijeli aplikaciju u tri nivoa: model, view, kontroler. Svaki od ovih dijelova obavlja određene akcije i preuzima odgovornost za specifične zadatke.

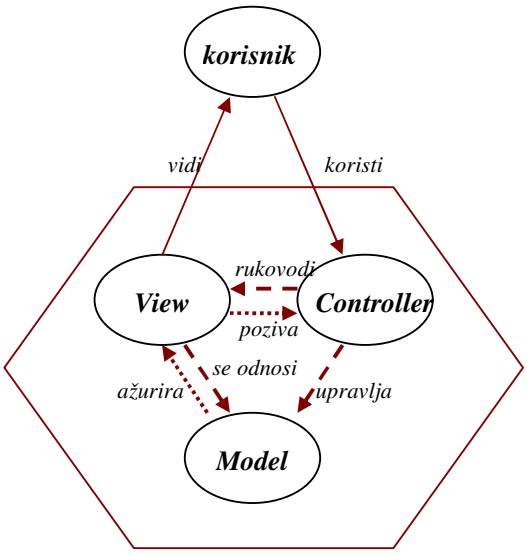
Model je jezgra aplikacije. On prezentira podatke i logiku ili operacije koje upravljaju pristupom i modifikacijom tih podataka. Često model predstavlja softversku aproksimaciju realnog svijeta. Model obavještava view-ove kada dolazi do promjena modela i osigurava upite view-a o njegovom stanju. Osim toga, omogućava kontroleru da pristupi funkcionalnostima aplikacije koje su sadržane modelom.

View predstavlja predstavu sadržaja modela. On pristupa podacima iz modela i specificira kako bi podaci trebali biti predstavljeni. Također, ažurira predstavu podataka kada dođe do promjene kod modela. View prosljeđuje korisnički ulaz kontroleru. Može postojati više od jednog view-a za isti model i svaki od tih view-a predstavlja sadržaj modela na drugoj površini, tj. displeju.

Kontroler definiše ponašanje aplikacije. On predstavlja korisničko okruženje predstavljeno korisniku da bi manipulisao aplikacijom. On obradi (završi) korisnički zahtjev i selektira view za prezentaciju, interpretira korisnički ulaz i mapira ga u akcije koje će izvršiti model. Kontroler selektira slijedeći view za prikaz u zavisnosti od korisničkih interakcija i izlaza operacija modela. Aplikacija obično ima jedan kontroler za jedan skup sličnih funkcionalnosti.

B. *Veze u MVC modelu*

Vidimo da su model, kontroler i view usko povezani. To znači da se moraju i referencirati jedni na druge. Slika 1. ilustrira osnovne Model-View-Kontroler odnose:



Slika 1. MVC odnosi i veze

Ova slika pokazuje osnove komunikacije između modela, view-a i kontrolera. Kao što slika prikazuje, model pokazuje na view, što predstavlja promjene koje model šalje view-u. Model ne treba da zna ništa o view-u kojeg posmatra, dok s druge strane, view tačno zna na koji se model odnosi. View ima pokazivač na model, omogućavajući mu da pozove bilo koju funkciju modela. Osim toga, view ima i tačkasti pokazivač na kontroler, s tim što ne poziva funkcije kontrolera definirane baznim klasama.

Kontroler ima pokazivač i na model i na view i poznaje tipove oba. S obzirom da kontroler rukovodi ovom triadom mora poznavati i tip modela i tip view-a da bi na odgovarajući način odgovorio na korisnički zahtjev.

C. *Zašto MVC?*

Odvajanjem odgovornosti između modela, view-a i kontrolera reducira se dupliciranje koda i sama aplikacija lakša je za održavanje. Pored toga, olakšava se upravljanje podacima, bilo kod dodavanja novih podataka ili mjenjanja njihovih prezentacija, jer se na ovaj način razdvaja logika razvoja od podataka.

MVC je posebno pogodan za interaktivne web aplikacije – aplikacije kod kojih web korisnik razmjenjuje informacije sa web stranicom, aplikacije sa velikim brojem prozora koji se ponavljaju, mnogostrukim

zahtjevima za podacima i njihovim prikazom, složene web aplikacije.

IV. Framework

A. Opis framework-a

Neka nam je zadatak da napravimo web aplikaciju. Inicijalni dizajn može biti prihvatljiv za tu jednu aplikaciju. Mogućnost generalizacije za više aplikacija se može postići samo izgradnjom tih aplikacija i uviđanjem zajedničkih karakteristika. Pravilo generalizacije je: napravi aplikaciju, napravi drugu aplikaciju koja je malo drugačija od prve, i konačno, napravi treću aplikaciju koja je malo drugačija od prve dvije. Ukoliko se sve tri aplikacije odnose na istu domenu, zajedničke karakteristike bit će vidljive. Ukoliko pravimo slijedeću aplikaciju sličnih osobina kao prethodne, onda je gubljenje vremena izgradnja te aplikacije od početka. Rješenje za ovakvu izgradnju aplikacija predstavlja framework.

Postoji više različitih definicija framework-a. Jedna od njih glasi: "Framework je re-upotrebljiv dizajn cijelog sistema ili jednog njegovog dijela koji je predstavljen skupom apstraktnih klasa i načinom na koji su povezane njihove instance." Druga definicija kaže: "Framework predstavlja kostur aplikacije koja se može prilagoditi različitim oblastima." Ove dvije definicije nisu protivrječne: prva opisuje strukturu framework-a, dok druga govori o njegovoj namjeni [27].

Dobar framework može značajno smanjiti cijenu razvoja aplikacije, jer dozvoljava re-upotrebu dizajna i koda. Nisu potrebne nove tehnologije, jer se može implementirati sa postojećim OO programerskim jezicima.

Razvoj dobrog framework-a je skupa investicija. Framework treba biti dovoljno jednostavan da se shvati, a pri tome da sadrži sve funkcionalnosti koje se mogu ili brzo iskoristiti ili po potrebi promijeniti tako da imaju slične svrhe.

Framework predstavlja samo jednu od re-upotrebljivih tehnika. Idealna re-upotrebljiva tehnika osigurava komponente koje se jednostavno mogu povezati da se dobije novi sistem. Čovjek koji razvija softver ne mora znati kako je komponenta implementirana, dok su karakteristike komponente lako shvatljive. Rezultujući sistem će biti efikasan, jednostavan za ugradnju i pouzdan. Sistem električne energije je nešto poput ovog: možete kupiti toster u jednoj prodavnici, a televizor u drugoj i oba uređaja će bez problema raditi u vašem stanu. Ljudi nemaju problema pri uključivanju tostera u struju, iako većina tih ljudi ne zna Ohmov zakon [27].

B. Osobine framework-a i njegova namjena

Frameworks su namjenjeni određenim poslovnim područjima (kao telekomunikacije ili procesiranje podataka) i aplikacijskim domenama (kao korisnički interface).

Osnovna prednost frameworka se ogleda u modularnosti, reupotrebljivosti i mogućnosti nadogradnje. Ta prednost ogleda se u osnovnim osobinama framework-a: *modulanrost*, *re-upotrebljivost*, i *nadogradnja.*

Modularity -- Modularnost frameworka nam pruža mogućnost poboljšanja kvaliteta softvera, lociranjem promjena u dizajnu i implementaciji. Ova lokalizacija umanjuje trud potreban za razumijevanje i održavanje postojećeg softvera.

Reusability -- Prikazivanje korištenih komponenti u vidu drveta omogućava programeru da lakše uvidi koje su to komponente koje se mogu iskoristiti i u drugim aplikacijama. Reupotreba komponenti frameworka dovodi do povećanja produktivnosti programera, kao i poboljšanja kvaliteta, performansi i interoperabilnosti softvera.

Extensibility -- Mogućnost nadogradnje je esencijalna stvar, koja omogućava prilagođavanje novim aplikacionim servisima. Time se obezbjeđuje dug život framework-a.

V. Razvoj web aplikacije za profesore i studente

A. Kratak opis aplikacije i namjena

Pomenuta aplikacija u ovom radu namijenjena je da olakša rad studentima i profesorima na predmetu Projektovanje sistemskog softvera (PSS).

Na početnoj stranici web aplikacije se nalaze linkovi za prijavu (Login), materijale, ispitne rokove, raspored predavanja, nastavni plan, često postavljena pitanja (Frequently Asked Questions - FAQ), opis aplikacije (About) i ostale informacije od značaja za studente.

Aplikacija se sastoji od početne stranice dostupne svim posjetiocima, profesorskog dijela i studentskog dijela na koje se korisnik usmjerava u zavisnosti od korisničke prijave. Postoje dva tipa korisnika, *profesor* i *student*. U zavisnosti kojem tipu pripada, nakon prijave korisnika *profesor* ili korisnika *student* otvaraju se različite forme.

Korisnik *profesor* ima mogućnost da unese ili izbriše studente, ukloni ili doda pojedina ispitna pitanja, pregleda rezultate koje su ostvarili studenti, doda FAQ, itd.

Korisniku *student* se, nakon prijave, otvara forma preko koje može pristupiti obrascu sa ispitnim pitanjima. Nakon rješavanja testa, studentu se prikazuje ostvareni rezultat (u postotcima) koji se automatski pamti tako da ga profesor može pogledati. Studenti na pomenutoj formi mogu naći listu korisnih linkova. Također, mogu ostvariti komunikaciju sa profesorom putem e-maila, ukoliko im je potrebna pomoć ili savjet.

B. Izvedba aplikacije i korišteni alati

Ova web aplikacija je napravljena koristeći Java [1, 3] programski jezik i mvcLite framework kojeg su studenti spiralno učili.

Osim toga, za izvedbu web aplikacije korišteni su:

- XML [2]
- HTML [5]
- Eclipse 3.0.1. [13]
- Tomcat 5.5.8. [14]
- Mozilla Firefox [15]
- Macromedia Dreamweaver MX
- Photoshop
- Sothink DHTML Menu 2005
- Java skripte

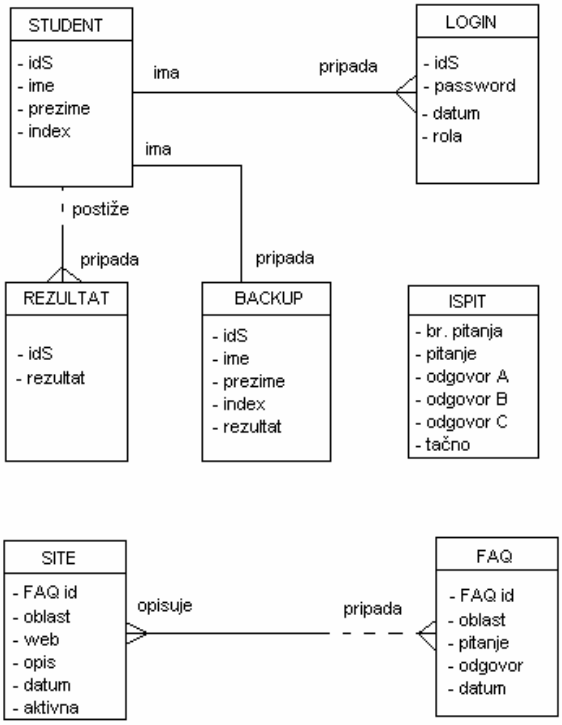
VI. OPIS WEB APLIKACIJE

A. *Entiteti i veze*

Aplikacija se sastoji od sedam entiteta:

- Login,
- Student,
- Rezultat,
- Site,
- Faq,
- Backup,
- Exam.

Između entiteta postoje tri veze od kojih je jedna many-to-many veza, a dvije 1:n. Veze između entiteta prikazane su na slici 2.



Slika 2. Entiteti i veze

Na slici 2. se može vidjeti da veza 1:n čini odnos student-rezultat, gdje student može imati više rezultata testa, tj. student može ponavljati test više puta, a rezultat jednog testa pripada samo jednom studentu. Drugu vezu čini odnos student-login.

Vezu m:n čini odnos site-faq. Za jedno pitanje iz određene oblasti može se unijeti više site-ova, a jedan site može biti od pomoći za više pitanja. Ovaj odnos se može vidjeti klikom na FAQ unutar menija prikazanog na početnoj stranici.

B. *Korisnici aplikacije*

Kao što je već navedeno, aplikacija je namijenjena profesorima i studentima na predmetu Projektovanje sistemskog softvera. Početna stranica koja je dostupna svim korinicima. Na stranici se nalaze opće informacije, materijali, ispitni rokovi itd. od značaja za sve studente, posjetioce te stranice.

Pored navedenog, na datoj stranici se također nalaze linkovi na FAQ i Login.

FAQ: ovaj link se odnosi na najčešća postavljena pitanja studenata, i odgovore profesora. Na studentskoj stranici se nalazi link na profesorov e-mail na koji studenti postavljaju upite profesoru. Kada se profesor prijavi, ima mogućnost da otvori formu za unos pitanja i odgovora koji će se izlistati na FAQ stranici.

Login: profesoru će biti dodijeljen *username* i *password* pomoću kojeg će se moći prijaviti na profesorski dio aplikacije. Nakon prijave, profesor ima mogućnost dodavanja novih studenata pri čemu definiše njihove *username*-e i *password*-e. Tako, svaki student može koristiti aplikaciju sa svojim *username*-om i *password*-om.

Profesorski dio aplikacije

Na web stranici aplikacije koja se odnosi na profesorski dio nalaze se linkovi za dodavanje/brisanje/modifikovanje studenata, ispitnih pitanja, FAQ-a, izlistavanje rezultata koje su postigli studenti i dodavanje i izlistavanje interesantnih linkova.

Studentski dio aplikacije

Kada se student prijavi za korištenje aplikacije, njegovo ime i prezime se prosljeđuje od jedne do druge studentske stranice. To je razlog što je početna stranica za studente stavljena je kao .jsp, a ne statička HTML stranica.

Statičke stranice su HTML stranice čiji se kôd ne mijenja nakon što se stranica postavi na server. HTML je jezik koji se koristi za izgradnju ovakvih statičkih stranica. Svaka stranica HTML kôda je napisana od strane dizajnera prije nego što se stranica postavi na server. Kada server dobije zahtjev za statičnom stranicom, on obradi zahtjev, nađe stranicu i pošalje je browser-u.

S druge strane, dinamičke web aplikacije se prikazuju na browser-u kroz stranice koje se dinamčki kreiraju na serveru. Kada se pošalje zahtjev serveru za prikazivanje određene stranice, ta se stranica na serveru procesira, obično sa podacima koji su uzeti sa stranice koje je unio korisnik i koji se ugrađuju u strukturu same stranice, i kao takva se prikazuje korisniku.

Postoje dva načina na koja se web stranica može učiniti interaktivnom (dinamičnom). Prvi je putem skripti (npr. Java Script) koje se unose u sam HTML kôd web stranice. Drugi način je koristeći programske mogućnosti servera na kome se nalazi web aplikacija. Dinamičke stranice najčešće se prepoznaju po imenu datoteke na kraju web adrese. Tako umjesto statičnog .html stoji dinamički nastavak .jsp , .php, .asp, .aspx, itd.

Na web stranici za studente se nalazi link na stranicu sa ispitnim pitanjima (Test) i ponuđenim odgovorima. Nakon rješavanja testa studentu se prikazuje njegov (ne)uspjeh na testu. Također, studentu se pruža mogućnost da sa ovog mjesta pristupi korisnim linkovima, ili da pošalje e-mail profesoru.

VII. OPIS IZDVOJENIH FUNKCIONALNOSTI

U ovom radu, kao interesantne karakteristike aplikacije, u nastavku će biti opisan način proslijeđivanja parametara s jedne stranice na drugu, kao i dodavanje studenta. Ostale funkcionalnosti unutar aplikacije su rađene na istom principu kao i gore navedene ili sa neznatnim razlikama.

A. *Prosljeđivanje parametara*

Na početnoj stranici se nalazi link za prijavu korisnika. Korisnici, studenti, prije prijave moraju biti uneseni u bazu podataka da bi mogli pristupiti studentskom dijelu aplikacije. Unos podataka vrši profesor. Ti parametri smješteni su u posebnu XML datoteku. Sve XML datoteke koje sadrže podatke o studentu su međusobno povezane jedinstvenim identifikatorom – idS.

Kada korisnik, prilikom prijave, unese svoje korisničko ime i lozinku, vrši se upoređivanje tih parametara sa onima koji su ranije uneseni.

Ukoliko se ispostavi da se radi o studentu koji se nalazi u bazi, preko već pomenutog parametra – idS, pretražuje se slijedeća XML datoteka. Na taj način se dolazi i do ostalih podataka o studentu (ime, prezime, index) koji će se proslijediti na stranice koje student dalje otvara.

Na isti način je povezana i XML datoteka koja sadrži rezultate testa koje su postigli studenti.

B. *Dodavanje studenata*

Na stranici namijenjenoj profesoru se, pored ostalih, nalazi i link koji omogućava dodavanje studenta (Add). Forma u koju profesor unosi nove podatke (idS, ime, prezime, index, username, password, datum i rola). Profesor inicijalno postavlja određeno korisničko ime i lozinku, dok student već nakon prve prijave ima mogućnost da promijeni lozinku.

Nakon pritiska na dugme *Add* provjerava se format unosa podataka (npr. datum treba biti unesen kao *dd.mm.gggg*.). Ako neki od podataka nije ispravno unesen, korisnik dobija informaciju o tome i mora ponovo unijeti pogrešno unesene podatke.

U slučaju da su podaci ispravno uneseni, dio podataka (idS, ime, prezime i index) se spašava u jednu XML datoteku, dok se ostali podaci, radi jednostavnosti, spašavaju u drugu. Ove dvije XML datoteke su povezane preko idS.

Da bi se spriječilo gubljenje podataka, svi uneseni podaci se unose u dodatnu XML datoteku u kojoj se podaci trajno čuvaju. Ova datoteka, na neki način, predstavlja arhivu podataka o prošlim i sadašnjim studentima. Ukoliko profesor izbriše nekog studenta, ti podaci će biti izbrisani iz prve dvije datoteke, dok će u ovoj ostati netaknuti..

VIII. ZAKLJUČAK

Razvoj dinamičkih web aplikacija je kompleksan i zahtjevan proces koji podrazumjeva znanje u različitim oblastima, kao što su framework, Java Server Pages (JSP), Servlet, zatim dizajn i programiranje, kao i korištenje različitih alata koji pomažu pri kreiranju web stranica. Osim toga važno je obratiti pažnju na strukturu stranice, hijerarhiju, organizovanost i jednostavnost koji će omogućiti korisniku da na brz i efikasan način dođe do traženih informacija. Moglo bi se reći da je projektovanje web stranica, u neku ruku, umjetnost koja u sebi obuhvata znanje, vještinu i talenat.

Kada je riječ o razvoju web aplikacija danas, najpopularnija kategorija software-a koja pruža veliku pomoć dizajnerima i povećava njihovu produktivnost jeste framework.

U projektu koji je razvijen u okviru predmeta PSS, osnovni cilj bio je razvoj dinamičke web aplikacije koristeći Java programski jezik i mvcLite framework. mvcLite je izučavan spiralno tako što su studenti počeli od najjednostavnijih zahtjeva vezanih za dizajn i projektovanje web aplikacija. Nakon kratkog vremena studenti su stekli znanje koje im je omogućilo rješavanje kompleksnijih projekata.

U procesu kreiranja web aplikacije za profesore i studente, vodilo se računa o funkcionalnosti web stranica i pogodnosti njihovog korištenja od strane korisnika (profesora i studenata). Osim ponuđenog, na web stranicu mogu biti dodani i drugi sadržaju i elementi. Aplikacija nije isključivo napravljena za odabrani predmet, te jednostavno može biti prilagođena različitim nastavnim predmetima.

REFERENCES

[1] B. Eckel, *Thinking in Java*, 2002.

[2] E.R. Harold, *Processing XML with Java,* 2002.

[3] F. Cohen, *Java Testing and Design*, 2004.

[4] J. Falkner, K. Jones, *Servlets and JavaServer Pages: The J2EE Technology Web Tier*, 2003.

[5] T. Stauffer, *HTML By Example*, 1999.

[6] D. Elderbrock, D. Karlins, *"FrontPage 2002 Biblija"*, 2002. Mikro knjiga.

[7] D. Ridjanovic, D., "Magicne kutije: Spiralni pristup ucenju Java programskog jezika i Swing grafičke biblioteke", http://drdb.fsa.ulaval.ca/mk/, 2004.

[8] Ridjanovic, D., "Learning Java in Spirals", 27th International Convention MIPRO 2004, Opatija, Croatia, May 24 - 28, 2004.

[9] Ridjanovic, D. and V. Okanovic, "Using Database Framework in Web Applications", IEEE MELECON 2004, Dubrovnik, Croatia, May 10- 14, 2004.

[10] Ridjanovic, D., V. Okanovic and B. Zaciragic, "Open Source Java Frameworks", XIX Medjunarodni Simpozij, Informacione i komunikacione tehnologije, B&H, Sarajevo, 2003-12-1.

[11] Ridjanovic, D., "Pedagogical Tools for Database Design", 2nd International Conference on the Principles and Practice of Programming in Java, Kilkenny City, Ireland, June 16-18, 2003.

[12] Ridjanovic, D., "Model-View-Controller Pattern based Web Application Design", XVIII Medjunarodni Simpozij, Informacione i komunikacione tehnologije, B&H, Sarajevo, 2001-11-26.

[13] http://www.eclipse.org/

[14] http://jakarta.apache.org/tomcat/index.html

[15] http://www.firefox.com

[16] http://drdb.fsa.ulaval.ca/mvcLite/

[17] http://st-www.cs.uiuc.edu/users/johnson/frameworks.html

[18] http://csis.pace.edu/~bergin/mvc/mvcgui.html

[19] http://www.hibernate.org/

[20] http://struts.apache.org/

[21] http://www.springframework.org/

[22] http://ootips.org/mvc-pattern.html

[23] http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/app-arch/app-arch2.html

[24] http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/web-tier/web-tier5.html

[25] http://www.objectarts.com/EducationCentre/Overviews/MVC.htm

[26] st-www.cs.uiuc.edu/users/johnson/cs329/2003/framework97.pdf

[27] http://www.cs.indiana.edu/~cbaray/projects/mvc.html

[28] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

[29] http://www.cs.wustl.edu/~schmidt/CACM-frameworks.html