RAZVOJ WEB i DESKTOP APLIKACIJE ZA EVIDENCIJU STUDENATA KORIŠTENJEM OBJEKTNOG PRISTUPA

Periša, Ivan

Master's thesis / Specijalistički diplomski stručni

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **The Polytechnic of Rijeka / Veleučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:125:634489

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-17**

Repository / Repozitorij:

Polytechnic of Rijeka Digital Repository - DR PolyRi

VELEUČILIŠTE U RIJECI

Ivan Periša

RAZVOJ WEB i DESKTOP APLIKACIJE ZA EVIDENCIJU STUDENATA KORIŠTENJEM OBJEKTNOG PRISTUPA

(specijalistički završni rad)

Rijeka, 2018.

VELEUČILIŠTE U RIJECI
Poslovni odjel
Specijalistički diplomski stručni studij Informacijske tehnologije u poslovnim sustavima

RAZVOJ WEB i DESKTOP APLIKACIJE ZA EVIDENCIJU STUDENATA KORIŠTENJEM OBJEKTNOG PRISTUPA

(specijalistički završni rad)

MENTOR

dr.sc. Ivan Pogarčić, prof.v.š

STUDENT

Ivan Periša, bacc.inf.

MBS: 2422000112/16

Rijeka, lipanj 2018.

VELEUČILIŠTE U RIJECI

Poslovni odjel

Rijeka, siječanj 2018.

ZADATAK
za specijalistički završni rad

Pristupniku Ivanu Periši **MBS: 2422000112/16**

Studentu specijalističkog stručnog studija Specijalistički diplomski stručni studij Informacijske tehnologije u poslovnim sustavima izdaje se zadatak završni rad – tema završnog rada pod nazivom:

RAZVOJ WEB i DESKTOP APLIKACIJE ZA EVIDENCIJU STUDENATA KORIŠTENJEM OBJEKTNOG PRISTUPA

Sadržaj zadatka: Prikazati način izrade i poduzete korake pri izradi web rješenja. Opisati aktivnosti u razvojnom okviru Laravel uz primjenu PHP jezika. Opisati razvoj desktop aplikacije uz primjenu programskog jezika Java. web development kao specifičnog načina pripreme i izrade informatičkih aplikacija orijentiranih web-u. Za obje aplikacije priložiti potrebnu dokumentaciju. Navesti potrebne alate i programe koji će pomoći u razvoju aplikacija. Objasniti potrebne komponente hardvera i softvera te primjenu operacijskih sustava i potrebnih programskih alata. Opisati i pojasniti potrebne dijelove objektno orijentiranog dizajna. Opisati način korištenja aplikacije. Opisati mogućnosti komercijalizacije proizvoda u određenim područjima djelatnosti.

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 20. siječnja 2018. **Predati do:** _____

Mentor:

dr.sc. Ivan Pogarčić, prof.v.š.

Pročelnik odjela:

mr.sc. Marino Golob, predavač

Zadatak primio dana: 20. siječnja 2018.

Ivan Periša, bacc.inf.

I Z J A V A

Izjavljujem da sam specijalistički završni rad pod naslovom **RAZVOJ WEB I DESKTOP APLIKACIJE ZA EVIDENCIJU STUDENATA KORIŠTENJEM OBJEKTNOG PRISTUPA** izradio samostalno pod nadzorom i uz stručnu pomoć mentora **dr. sc. Ivana Pogarčića.**

Ivan Periša

Ivan Periša

(potpis studenta)

Sažetak:

Evidentiranje studenata ima važnu ulogu u poticanju dolaska na nastavu, jer studenti dobiju korisne informacije i smanje količinu vremena samostalnim učenjem. Evidentiranje studentske prisutnosti se odvija na različite načine (pomoću potpisa studenta, prozivanje studenata, razni sustavi npr. Merlin) ovisno o visokoobrazovnoj ustanovi. Kroz ovaj rad prikazan je i opisan sustav za evidenciju studenata putem desktop i web aplikacije koji ujedinjuje sve načine evidentiranja studentske prisutnosti. Sustav je napravljen u dvije aplikacije (desktop i web) radi lakšeg evidentiranja studentska prisutnosti. Desktop aplikacija koristi magnetni čitač kartica (X-ica) za automatsko evidentiranje prisutnosti studenata što je ujedno i glavna svrha ovog sustava, dok na web aplikaciji evidentiranje prisutnosti se obavlja ručno. Web aplikacija je izrađena u razvojnom okviru Laravel (programski jezik PHP), a desktop aplikacija u programskom jeziku Java. Obje aplikacije komuniciraju s istom bazom podataka (relacijskom bazom podataka). Sustav omogućava pregled evidencije studenata u bilo kojem trenutku, te je izrađen da se jednostavno implementira na različitim visokoobrazovnim ustanovama.

Ključne riječi: UML, Laravel, evidencija, prisutnost, studenti

SADRŽAJ

1. Uvod 1
2. Opis sustava 2
2.1. Desktop aplikacija 3
2.2. Web aplikacija 3
2.2.1. Razvojni okvir Laravel 4
3. Analiza sustava 5
3.1. Dijagram slučajeva korištenja (*Use Case Diagram*) 5
3.2. Slučaj korištenja - unos, izmjena i brisanje kolegija 9
3.3. Slučaj korištenja - evidentiranje studenta preko čitača kartica 12
4. Grafički dizajn aplikacija 14
4.1. Dizajn desktop aplikacije 14
4.2. Dizajn web aplikacije 18
5. Kreiranje modela podataka 21
5.1. UML dijagram klasa 21
5.2. EVA model podataka 24
5.3. Relacijski model podataka 27
6. Objektno orijentirani dizajn 29
6.1. Sekvencijalni dijagram – unos, izmjena i brisanje kolegija 30
6.2. Sekvencijalni dijagram – evidentiranje studenata preko čitača kartica 33
7. Implementacija sustava u Laravel razvojnom okruženju 34
8. Prikaz uporabe programskog rješenja 38
9. Zaključak 44
Literatura 45
Popis slika 46

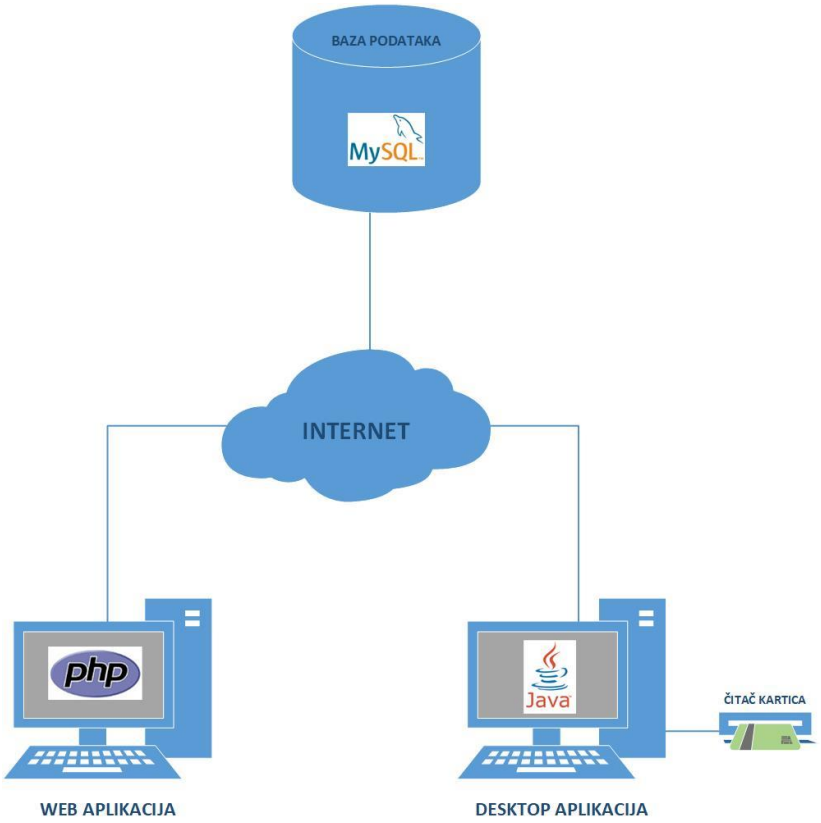
1. Uvod

Evidentiranje studenata na nastavi potiče studente da dobiju korisne informacije i smanje količinu vremena provedenog samostalnim učenjem i istraživanjem. Na Veleučilištu u Rijeci način provođenja evidencije studenata odvija se na nekoliko načina. Neki profesori koriste listove papira na koje se studenti potpisuju, neki usmeno prozivaju studente, dok nekolicina njih koristi sustav „Merlin“. Studenti koji dolaze na nastavu (vježbe i predavanja) mogu dobiti dodatne bodove od 0 do 6% ocjene ovisno koliko su puta bili na predavanju od ukupne satnice. Cilj sustava je ujediniti sve načine evidentiranja studenata. Glavna karakteristika mu je čitač kartica koji radi na principu da student provuče karticu (X-icu) kroz čitač kartica i na temelju podataka iz kartice evidentira svoju prisutnost. U nastavku rada dokumentirat će se izrada sustava za evidenciju studenata kroz dvije aplikacije (desktop i web) koje komuniciraju s centralnom bazom podataka. Krenut će se s opisom sustava, zatim analizom sustava gdje će se obraditi pojedini slučajevi korištenja i za njih dijagrami. Napraviti će se dijagrami klasa, EVA model te relacijski model za kreiranje baze podataka. Zatim slijedi objektno orijentirani dizajn koji će biti prikazan sekvencijalnim dijagramima, a nakon svega gotova grafička sučelja obje aplikacije.

2. Opis sustava

Svrha ovog sustava prije svega namijenjena je profesorima za lakše evidentiranje studentske prisutnosti na nastavi. Sustav je osmišljen na način da profesor može bilježiti studente preko dvije aplikacije (desktop i web) koje su spojene na jednu bazu podataka (relacijsku bazu podataka). Ukoliko profesor odabere bilježenje studenata putem desktop aplikacije (kod desktop aplikacije evidentiranje studenata vrši se putem čitača kartica) uvijek može na web aplikaciji dodati studente ukoliko je netko od njih zaboravio svoju karticu (*X-icu*). Profesor može kombinirati obje aplikacije za evidenciju studenata, odnosno može evidentirat studente na svakoj zasebno. Obje aplikacije imaju dva sučelja: administratorsko i profesorsko.

Slika 1. Prikaz arhitekture sustava



Izvor: Autor

2.1. Desktop aplikacija

Za izradu aplikacije korišten je objektno orijentirani programski jezik Java. Aplikacija je razvijena u besplatnom programu „NetBeans IDE“. U nastavku će se detaljno opisat funkcionalnost aplikacije.

Prilikom evidentiranja studenata preko desktop aplikacije, profesor se prije svega mora prijaviti u sustav sa svojim korisničkim imenom (*e-mailom*) i lozinkom. Nakon uspješne prijave u sustav profesor testira aplikaciju je li spojena sa čitačem kartica (čitač mora biti priključen u računalo preko USB - *Universal Serial Bus* porta) i bazom podataka kako bi mogao izabrati kolegij, termin i vrstu predavanje. Profesor može evidentirati studente samo za termin koji je na taj dan (npr. ako je datum 21.05.2018 profesor može evidentirati studente za taj termin samo toga dana) i u određeno vrijeme (ako termin traje npr. od 15:00 do 17:30, profesor može evidentirati studente samo u tome vremenskom razdoblju). Ukoliko je sva validacija uspješna, studenti se mogu evidentirati. Bilježenje studenata vrši se pomoću studentske kartice (*X-ice*) na način da svaki student provlači svoju karticu kroz čitač kartica nakon čega se bilježi prisutnost za svakog pojedinog studenta. Profesor može i obrisati studenta s termina prisustva. Osim bilježenja evidencije u profesorskom sučelju imaju još i opcije unosa termina za kolegije profesora, brisanje termina, prikaz studenata po terminu, te pregled studenata po kolegiju.

Drugi dio desktop aplikacije je administratorsko sučelje koje zahtjeva prijavu administratora u sustav. Administrator ima drugačije privilegije od profesora. On može unositi, izmjenjivati i brisati kolegije, unositi termine za kolegije, unositi nositelja kolegija, unositi studente na kolegij. Osim toga može pregledavati kolegije po profesoru, pregledavati studente po kolegiju i pregledavati studente po određenom terminu kolegija.

2.2. Web aplikacija

Aplikacija je izrađena u programskom jeziku PHP u razvojnom okviru Laravel. Program koji se koristio pri izradi web aplikacije je „PhpStorm“. U nastavku slijedi detaljan opis funkcionalnosti web aplikacije, te razvojnog okvira Laravel.

Web aplikacija za razliku od desktop aplikacije nema mogućnost automatske evidencije studenata preko čitača kartica. Evidencija studenata vrši se ručno na način da se odaberu studenti koji su prisutni na predavanju, te se evidentira njihovo prisustvo. Evidentiranje studenata za zadani termin može se mijenjati, što znači da profesor može evidentirati dodatno studente za taj termin za razliku od desktop aplikacije gdje se studenti ne mogu dodavati ručno. Web aplikacija u odnosu na desktop aplikaciju ima više operacija. Administrator ima ovlasti za registraciju novih korisnika (studenata, profesora, administratora), može povezivati studente i profesore s kolegijima, pregledavat studente po kolegijima, te pregledavat kolegije po profesorima. Administrator unosi nove studije fakulteta, uređuje ih, te ih briše. Također iste operacije može raditi i za godine studija, kolegije, razine pristupa. Što se tiče evidencije studenata, administrator može unositi termine za kolegije, pregledavat studente po terminu, dodatno bilježiti studente na termin nastave, brisati studente s termina, te pregledava ukupnu prisutnost studenata po kolegiju.

Profesor u web aplikaciji ima drugačije ovlasti od administratora. Profesor može registrirat studenta u sustav. Nadalje može pregledavati studente po svojim kolegijima, može unositi studente na svoje kolegije, pregledavati podatke studenata, te brisati studente s kolegija. Za svoje kolegije može unositi termine nastave, bilježiti studente na termin nastave, brisati studente s termina, pregledavati studente po terminu i pregledavati ukupnu prisutnost studenata po kolegiju.

2.2.1. Razvojni okvir Laravel

Laravel je besplatan razvojni okvir otvorenog koda za PHP programski jezik koji je namijenjen za razvoj web aplikacija. 2015. godine proglašen je jednim od najpopularnijih PHP razvojnih okvira u svijetu. Poznat je po jednostavnosti korištenja, izražajne sintakse, jasne strukture i jako detaljne dokumentacije što uvelike korisnicima olakšava rad. Koristi MVC (*Model-View-Controller*) pristup jer je i napravljen s ciljem razvoja aplikacija pomoću MVC-a. MVC dijeli aplikaciju u prezentacijski sloj (pogledi), sloj poslovne logike (upravljači) i podatkovni sloj (modeli). Preporuča se instalacija alata *Composer* koji upravlja zavisnostima u PHP-u. Instalira i ažurira biblioteke o kojima projekt ovisi. Ne upravlja paketima kao standardni upravitelj paketima, već ih instalira unutar direktorija projekta pri svakom stvaranju novog

projekta. Laravel dolazi s dosta ugrađenih alata, a najpoznatiji su: *Middleware, Blade, Artisan, Eloquent* ORM (*Object Relational Mapper*).

Međusoftver (*Middleware)* je softver koji djeluje između aplikacije i mreže. Ima mehanizam koji filtrira HTTP (*HyperText Transfer Protocol*) zahtjeve koji dolaze aplikaciji. Međusoftver je dakle zadužen za provjeru je li korisnik prošao autentikaciju. Ukoliko se utvrdi suprotno, međusoftver će preusmjeriti korisnika na pogled za prijavu. Blade je alat za predloške pogleda (*views*). Za razliku od drugih alata ne zabranjuje korištenje običnog PHP koda u pogledima. Ovaj alat uvelike umanjuje posao izrade *front-end* koda i pruža bolju funkcionalnost. Glavna prednost alata Blade je stvaranje sekcija i nasljeđivanje predložaka. Artisan je PHP sučelje naredbenog retka. Pruža brojne korisne naredbe koje uvelike pomažu pri izradi aplikacije (naredbe za rad s bazom podataka, praćenje rada na aplikaciji, upravljanje sesijama i događajima). Eloquent ORM služi za upravljanje bazom podataka. Svaki *Eloquent* model predstavlja jednu tablicu u bazi podataka, odnosno svakoj tablici pripada jedan model uz pomoć kojega komunicira s istom. Izvorni kod Laravela se nalazi na *GitHub* web stranici i licenciran je pod uvjetima *MIT License*.

3. Analiza sustava

Analiza sustava primjenjuje se u najranijoj fazi razvijanja informacijskog sustava gdje služi za prikaz globalne strukture. U fazi analize koriste se standardna sredstva za prikazivanje strukture pojedinih elemenata sustava. U ovom slučaju koristit će se UML (*Unified Modeling Language*) dijagrami. U ovoj fazi sustav se raščlanjuje na sastavne dijelove gdje se utvrđuje njegova struktura.

3.1. Dijagram slučajeva korištenja (*Use Case Diagram*)

Dijagram slučajeva korištenja opisuje funkcionalne zahtjeve sustava promatranih „izvana“. Prikazuje odnos između sudionika (aktera) i slučajeva korištenja. Kreira se u ranim fazama oblikovanja (najčešće kao prvi dijagram). Prikazuje što sustav treba raditi, a ne kako.

Dijelovi dijagrama slučajeva korištenja su:

- akteri (korisnici sustava, nisu njegov sastavni dio nego su u nekakvoj interakciji sa sustavom)
- veze (aktivnosti komunikacije aktera i slučajeva korištenja)
- slučaji korištenja (sekvence akcija koje su karakteristične za korištenje sustava, a predstavljaju funkcije koje sustav obavlja).

Aktere je potrebno imenovati ispod grafičkog simbola, a slučajeve korištenja unutar grafičkog simbola u obliku elipse (Slika 2.). Dijagram slučajeva korištenja ovog sustava za web aplikaciju ima dva aktera: administratora i profesora.

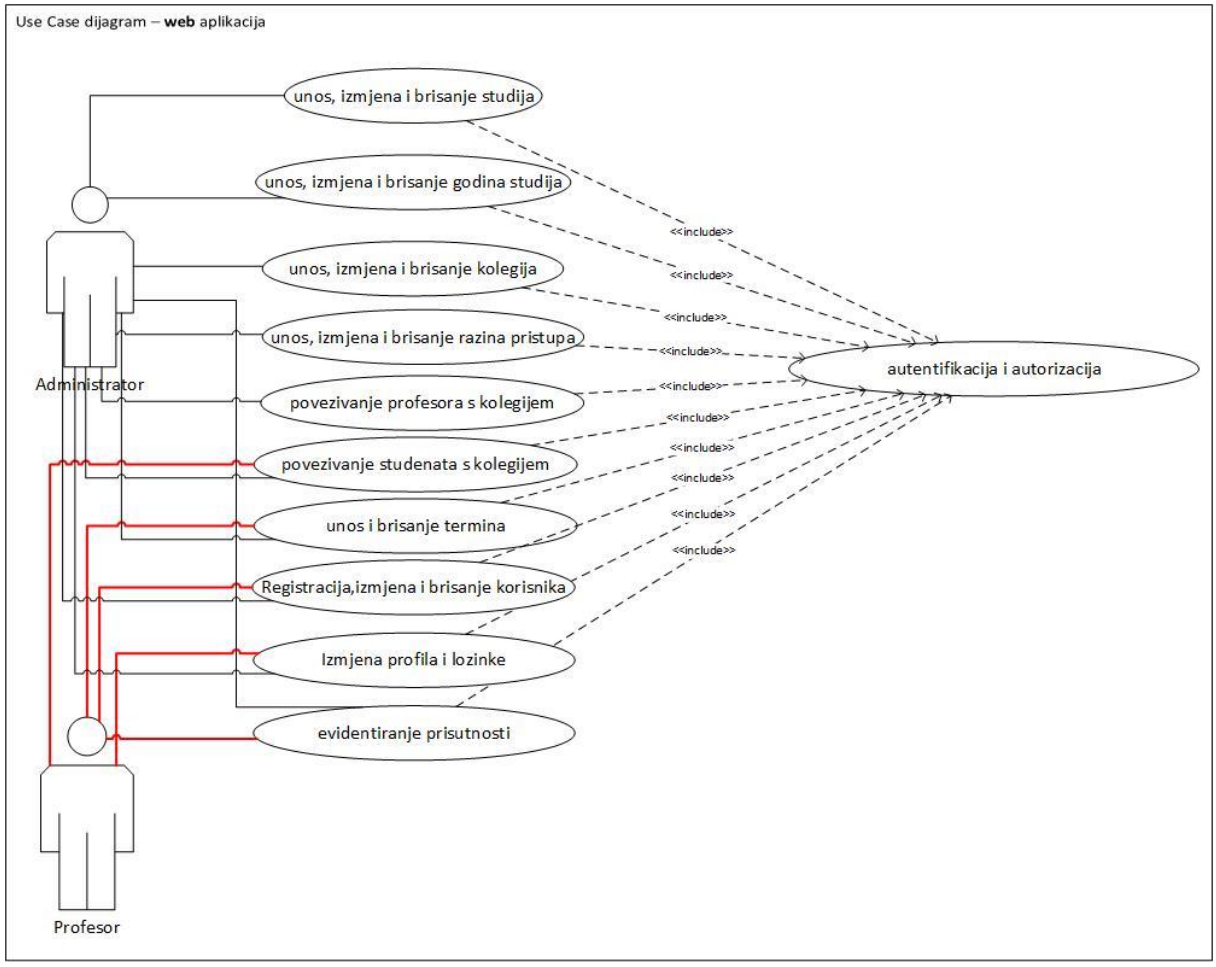
Administrator :

- Unos, izmjena i brisanje studija
- Unos, izmjena i brisanje godine studija
- Unos, izmjena i brisanje kolegija
- Unos, izmjena i brisanje razine pristupa
- Povezivanje profesora s kolegijem
- Povezivanje studenta s kolegijem
- Unos i brisanje termina
- Registracija i izmjena podataka korisnika
- Izmjena profila i lozinke
- Evidentiranje prisutnosti

Profesor :

- Povezivanje studenta s kolegijem
- Unos i brisanje termina
- Registracija i izmjena podataka korisnika
- Izmjena profila i lozinke
- Evidentiranje prisutnosti

Slika 2. Prikaz dijagrama slučajeva korištenja web aplikacije



Izvor: Autor

Drugi dijagram slučajeva korištenja ovog sustava se odnosi na desktop aplikaciju (Slika 3). Dijagram prikazuje aktere sustava desktop aplikacije i njihove slučajeve korištenja. Akteri sustava desktop aplikacije su: administrator, profesor i student.

Administrator :

- Unos, izmjena i brisanje kolegija
- Unos termina
- Povezivanje profesora s kolegijem
- Povezivanje studenta s kolegijem
- Evidentiranje prisutnosti

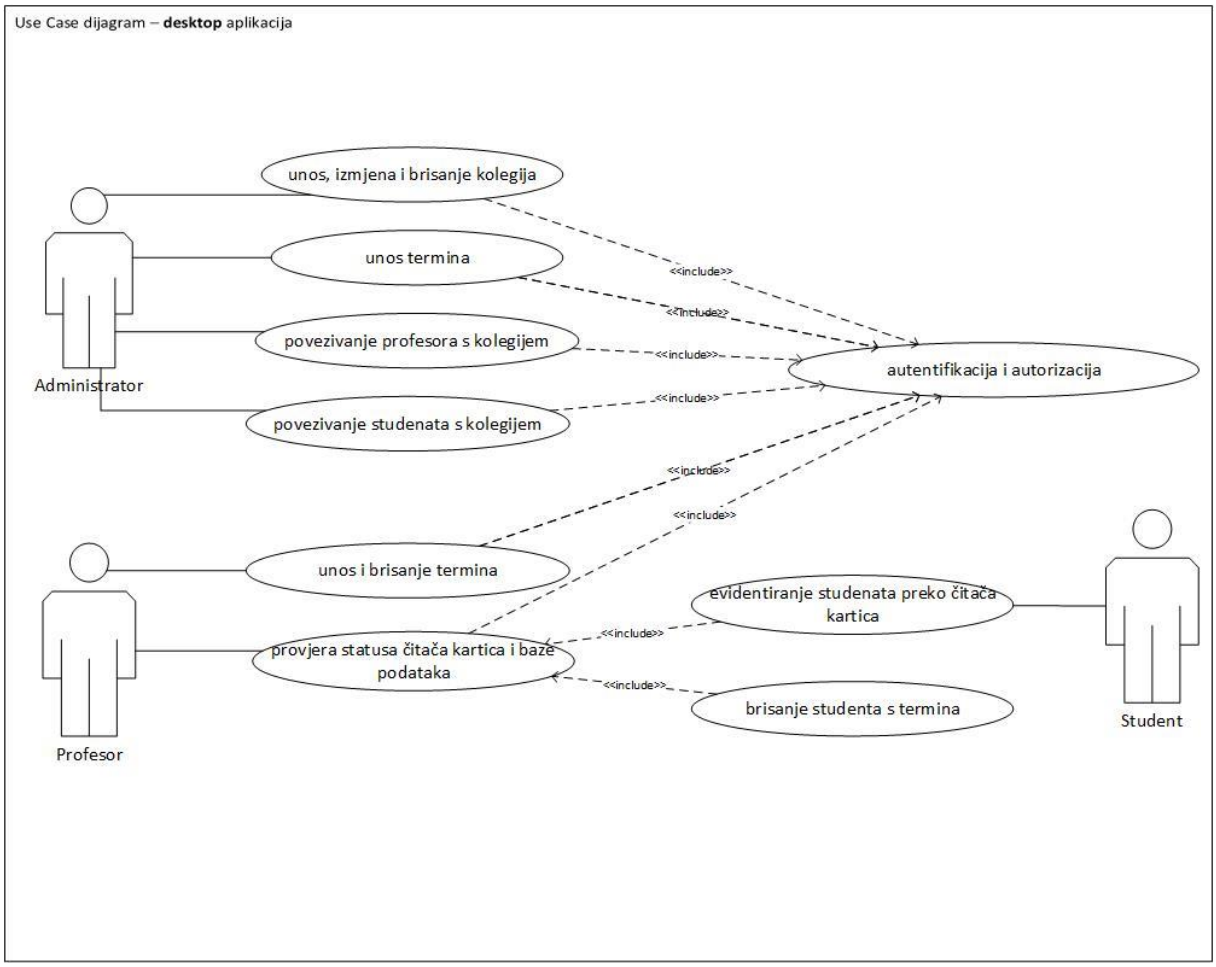
Profesor :

- Unos i brisanje termina
- Provjera statusa čitača kartica i baze podataka
- Brisanje studenta s termina

Student:

- Evidentiranje studenta preko čitača kartica

Slika 3. Prikaz dijagrama slučajeva korištenja desktop aplikacije



Izvor: Autor

Dijagrami aktivnosti prikazuju softversku funkcionalnost sustava iz unutrašnje perspektive sustava. Ovi dijagrami ne prikazuju sudionike sustava niti vanjsko sučelje prema krajnjim korisnicima. Prikazuju proceduralnu logiku, poslovni proces i radni tok aktivnosti koje se izvršavaju u sustavu korak po korak. Dijagrami aktivnosti stavljaju naglasak na jednostavnost te poslovne operacije koje se odvijaju slijedno, jedna za drugom. Svrstavaju se u dinamičke dijagrame jer razrađuju ponašanje sustava u smislu prijelaza između stanja.

U nastavku će biti detaljno opisani i prikazani dijagrami aktivnosti za pojedine slučajeve korištenja ovog sustava.

3.2. Slučaj korištenja - unos, izmjena i brisanje kolegija

Identifikacijski sažetak: Administrator unosi, izmjenjuje i briše kolegij.

Akteri: Administrator

Preduvjeti: Ispravni korisnički podaci. Kod unosa kolegija preduvjet je da isti nije prethodno unesen za određeni studij i godinu studija, a kod izmjene i brisanja kolegija preduvjet je da je taj određeni kolegij već unesen.

Tijek događaja:

Glavni uspješni scenarij (G):

1. Administrator odabire zahtjev za unos, izmjenu i brisanje kolegija.
2. Administrator izabire jednu od opcija(unos, izmjena ili brisanje).
3. Otvara se web stranica za unos, izmjenu ili brisanje kolegija.
4. Administrator unosi podatke za kolegij, izmjenjuje podatke o kolegiju ili briše kolegij.
5. Administrator sprema, izmjenjuje ili briše kolegij.
6. Sustav javlja da je kolegij unesen, izmijenjen ili obrisan.

Alternativne sekvence (A):

A1: Potvrda brisanja kolegija

4. Administrator briše kolegij.
5. Otvara se potvrdni prozor za brisanje kolegija.
6. Administrator odustaje od brisanja kolegija.
7. Povratak na G3.

Sekvence s greškom (E):

E1: Provjera validacije podataka

4. Administrator unosi ili izmjenjuje tražene podatke.
5. Sustav provjerava validaciju podataka.
6. Sustav javlja da podatci nisu dobro uneseni.
7. Povratak na G3.

E2: Provjera korištenja kolegija

4. Administrator izmjenjuje ili briše kolegij.
5. Sustav provjerava korištenje kolegija.
6. Sustav javlja da se kolegij koristi te se ne može izmijeniti ili obrisati .
7. Povratak na G3.

E3: Provjera identičnosti kolegija

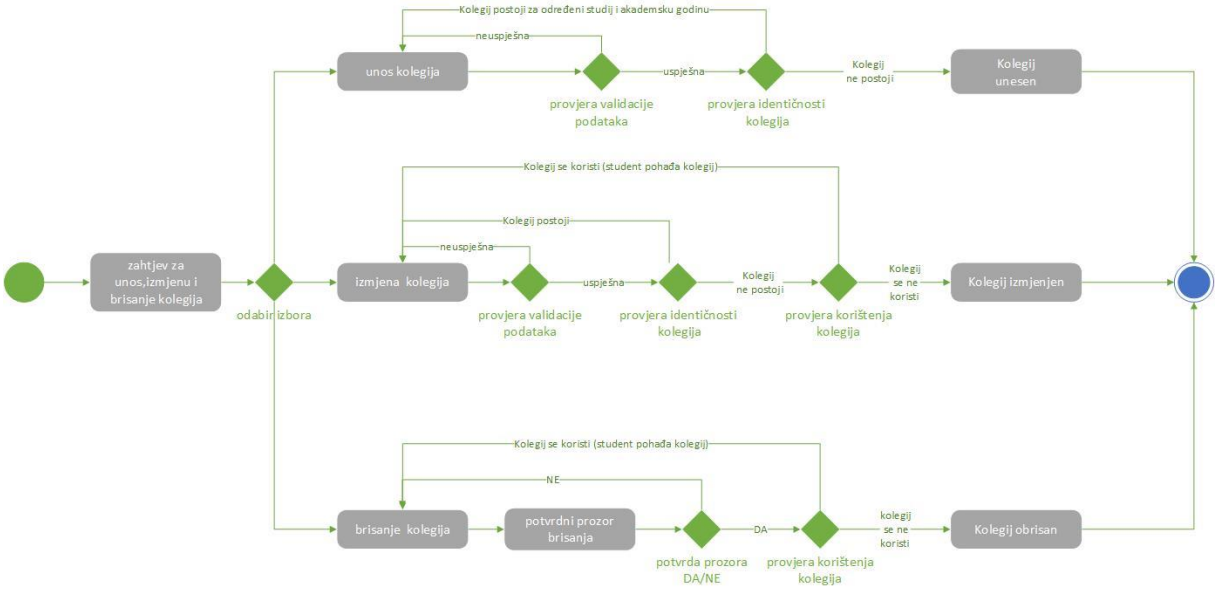
4. Administrator unosi ili izmjenjuje kolegij
5. Sustav provjerava postojanje unesenih podataka kolegija.
6. Sustav javlja da već postoji identični kolegij s tim podatcima.
7. Povratak na G3.

Ulazno-izlazni zahtjevi: računalo, tipkovnica, monitor, modem

Nefunkcionalni uvjeti (ograničenja): Aplikacija mora administratoru prikazati zatražene podatke, proces verifikacije korisničkih podataka mora biti vrlo pouzdan, promjene se moraju brzo ažurirati i prikazati

Po ovoj shemi dijagrama aktivnosti za slučaj korištenja „unos, izmjena i brisanje kolegija“ rade se na isti način i dijagrami aktivnosti za slučajeve korištenja: „Unos, izmjena i brisanje studija“, „Unos, izmjena i brisanje godine studija“, „Unos, izmjena i brisanje razine pristupa“ i „Unos i brisanje termina“

Slika 4. Prikaz dijagrama aktivnosti za unos, izmjenu i brisanje kolegija



Izvor: Autor

3.3. Slučaj korištenja - evidentiranje studenta preko čitača kartica

Identifikacijski sažetak: Student provlači karticu (X-icu) kroz čitač kartica

Akteri: Profesor, student

Preduvjeti: Čitač kartica spojen na računalo i komunicira s aplikacijom. Profesor odabire kolegij, termin za taj kolegij i vrstu predavanja.

Tijek događaja:

Glavni uspješni scenarij (G):

1. Profesor postavlja automatsku evidenciju studenata preko čitača kartica
2. Student provlači karticu (X-icu) kroz čitač kartica.
3. Sustav čita podatke s kartice i obrađuje iste
4. Sustav provjerava upis studenta na kolegij
5. Sustav provjerava evidenciju studenta
6. Student uspješno evidentiran

Sekvence s greškom (E):

E1: Provjera čitanja podataka

3. Sustav čita podatke s kartice i obrađuje iste.
4. Sustav provjerava podatke.
5. Sustav javlja da podatci nisu uspješno pročitani s kartice (X-ice).
6. Povratak na G2.

E2: Provjera validacije upisa na kolegij

4. Sustav provjerava upis studenta.
5. Sustav provjerava podatke o upisu.
6. Sustav javlja da student nije upisan na kolegij
7. Povratak na G2.

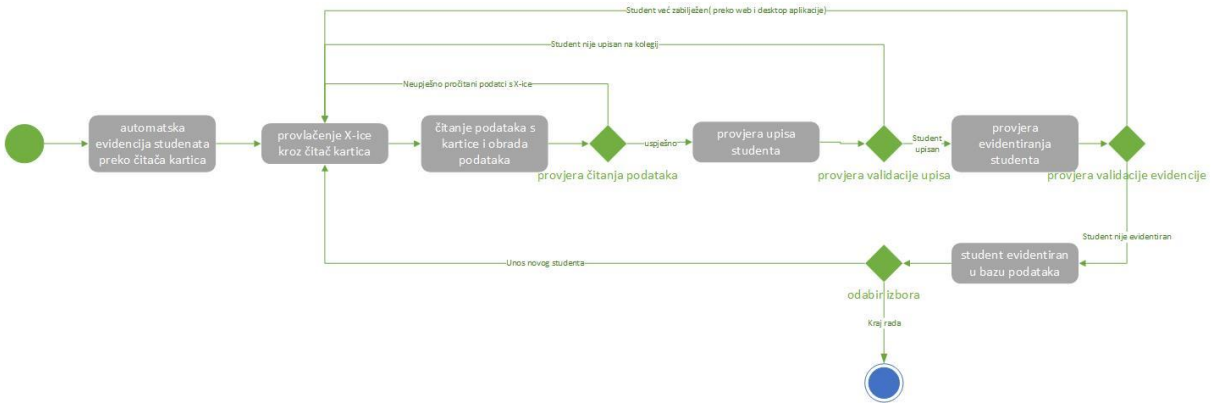
E3: Provjera validacije evidencije studenta

5. Sustav provjerava evidenciju studenta.
6. Sustav provjerava da li je student evidentiran za termin na koji se prijavljuje.
7. Sustav javlja da je student već evidentiran na taj termin
8. Povratak na G2.

Ulazno-izlazni zahtjevi: računalo, tipkovnica, monitor, modem

Nefunkcionalni uvjeti (ograničenja): aplikacija mora profesoru ili studentu prikazati zatražene podatke unutar dvije sekunde, promjene se moraju brzo prikazati.

Slika 5. Prikaz dijagrama aktivnosti za evidentiranje studenata preko čitača kartica



Izvor: Autor

4. Grafički dizajn aplikacija

U ovom poglavlju prikazat će se modeli desktop i web aplikacije, odnosno kako bi obje aplikacije trebale izgledati u produkcijskoj okolini.

4.1. Dizajn desktop aplikacije

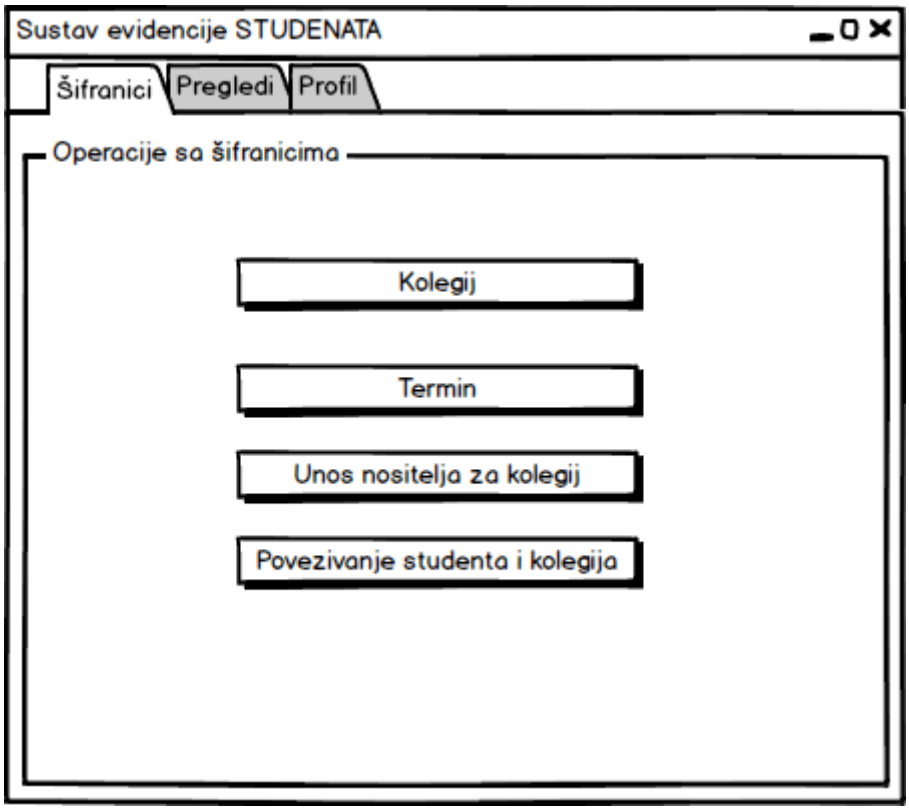
Sljedeća slika (Slika 6) prikazuje početno sučelje desktop aplikacije. Na početnom sučelju potrebno je unijeti podatke za prijavu u sustav. Korisnik nakon toga unosi podatke i treba stisnuti prijava. Sustav će korisniku ako su podatci ispravni otvoriti profesorsko ili administratorsko sučelje ovisno o ovlastima koje korisnik posjeduje.

Slika 6. Prikaz početnog sučelja desktop aplikacije

Izvor: Autor

Korisnik se prijavljuje u sustav sa svojim korisničkim podatcima. Sustav otvara početno administratorsko sučelje na temelju ovlasti korisnika (Slika 7). Administrator na početnoj stranici može vršiti operacije s kolegijima i terminima, odnosno može povezivati studenta i kolegij te profesora i kolegij (unositi nositelja za kolegij).

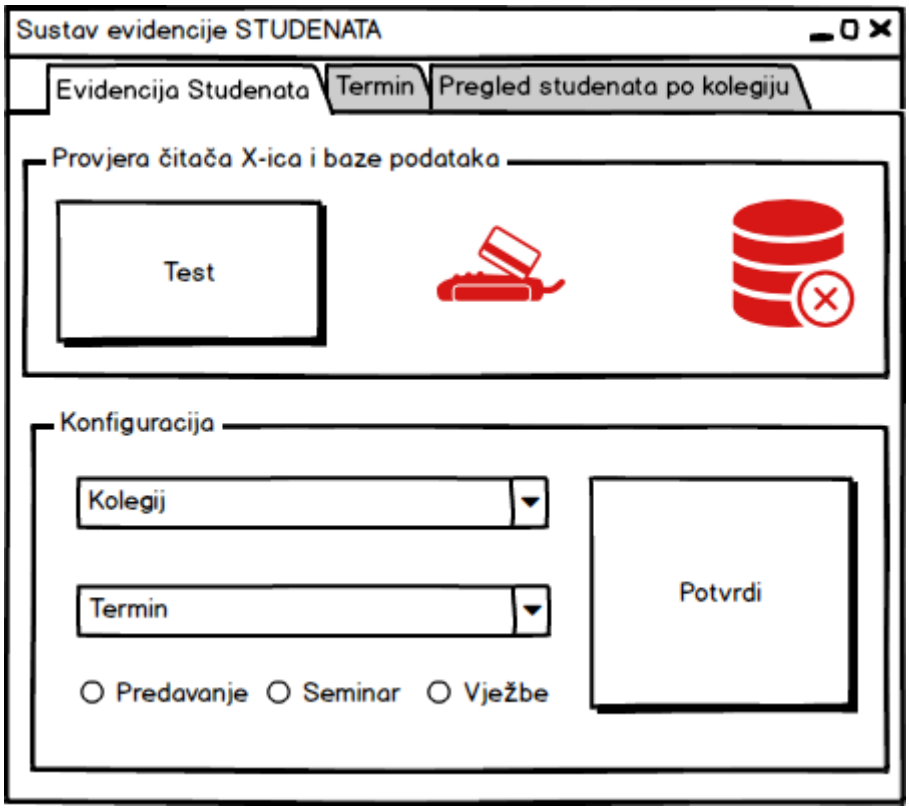
Slika 7. Prikaz početnog administratorskog sučelja desktop aplikacije



Izvor: Autor

Nakon uspješne prijave profesora u sustav otvara se profesorsko sučelje (Slika 8). Na početnoj stranici nalaze se dvije sekcije. Prva sekcija služi za provjeru povezanosti desktop aplikacije s čitačem kartica i bazom podataka. Na pritisak tipke „Test“ ikone baze podataka i čitača kartica se trebaju iz crvene boje promijeniti u zelenu ukoliko su povezani s aplikacijom. Druga sekcija ovisi o prvoj. Ukoliko test povezivanja nije uspješan ne može se prijeći na drugu sekciju. U drugoj sekciji namješta se konfiguracija za evidentiranje studenata, tj. odabire se kolegij, termin za taj kolegij i vrsta predavanja. Nakon odabira potrebno je pritisnuti „Potvrdi“.

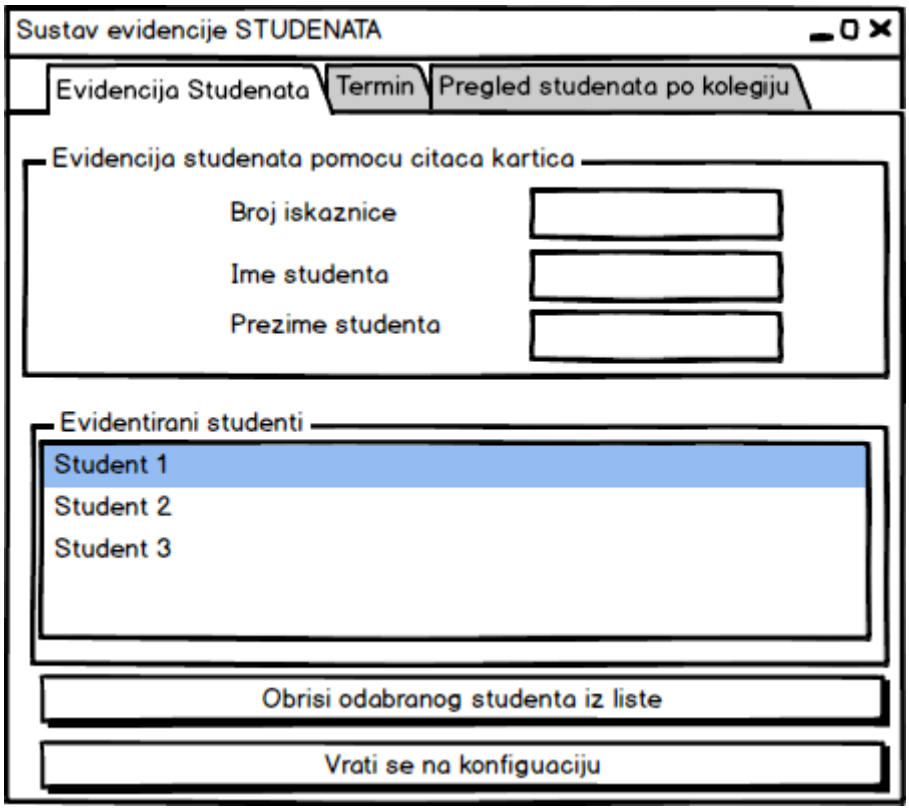
Slika 8. Prikaz početnog profesorskog sučelja desktop aplikacije



Izvor: Autor

Na tipku „Potvrdi“ s prethodne slike (Slika 8) otvara se sučelje za evidentiranje studenata. Nakon provlačenja kartice (X-ice) studenta kroz magnetski čitač kartica, automatski se popunjavaju polja: broj iskaznice, ime studenta i prezime studenta. Osim popunjavanja polja student koji je provukao karticu, također se automatski dodaje u listu evidentiranih studenata. Profesor može obrisati studenta s liste ako vidi neke nepravilnosti oko istog. Ukoliko je profesor prilikom konfiguracije nešto krivo odabrao kao npr. vrstu predavanja, može se vratit na pomoću gumba „Vrati se na konfiguraciju“ (Slika 9).

Slika 9. Prikaz sučelja za evidentiranje studenata na desktop aplikaciji



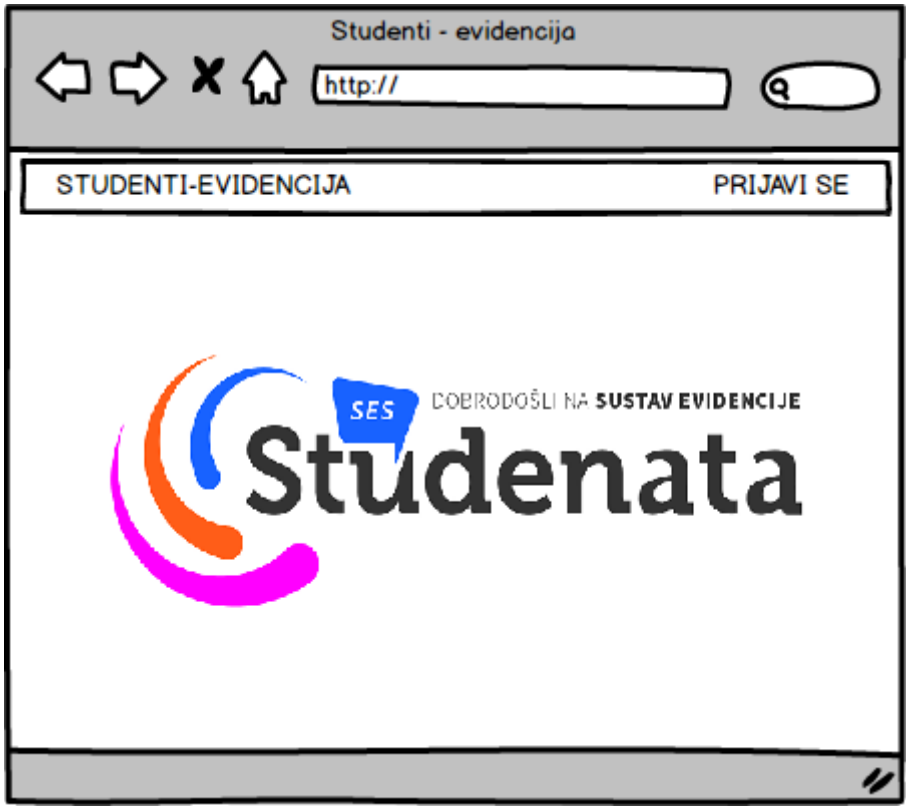
Izvor: Autor

U profesorovom sučelju osim evidentiranja studenata postoji sekcija za termin i sekcija za pregled studenata po kolegiju što je vidljivo na prethodnoj slici (Slika 9). U sekciji „Termin“ može pregledavati studente po terminu nastave, unositi termin i brisati termin.

4.2. Dizajn web aplikacije

Početna stranica web aplikacije prikazana je na sljedećoj slici (Slika 10). Stranica prikazuje sliku aplikacije. Za prijavu u sustav potrebno je kliknuti na „Prijavi se“ gdje korisnik unosi svoje podatke i prijavljuje se u sustav.

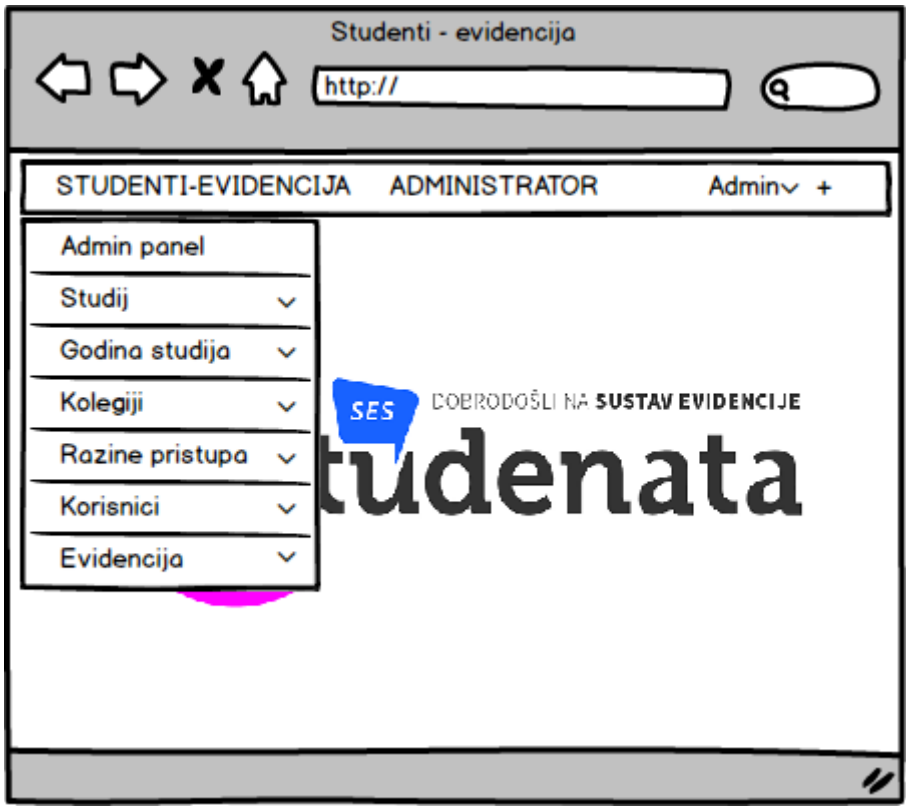
Slika 10. Prikaz početnog sučelja web aplikacije



Izvor: Autor

Nakon prijave u sustav otvara se aplikacijsko sučelje ovisno o tome koje ovlasti ima korisnik. U ovom slučaju otvara se administratorsko sučelje. Administrator u lijevom gornjem kutu ima izbornik gdje može izabrati željene radnje (operacije sa studijima, godinama studija, kolegijima, razinama pristupa, korisnicima i evidencijama (Slika 11).

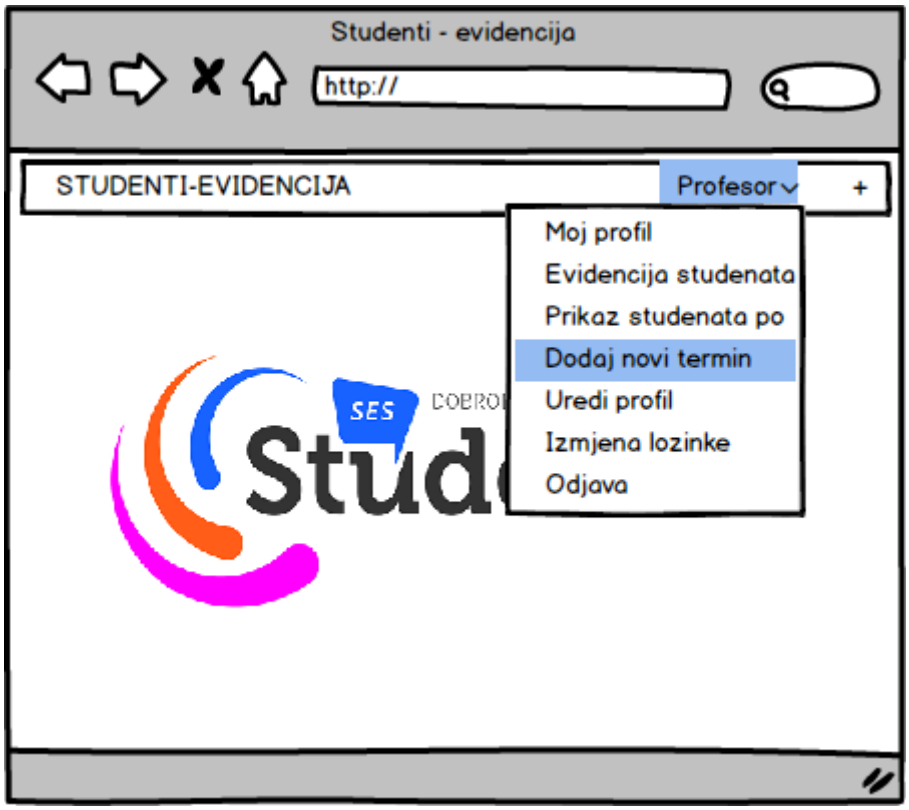
Slika 11. Prikaz početnoga administratorskog sučelja web aplikacije



Izvor: Autor

Prijavom u sustav kao profesor otvara se profesorsko sučelje (Slika 12). Za razliku od administratora profesor nema u lijevom kutu izbornik, već ima strelicu kraj svog imena. Na klik strelice otvara se izbornik profesora. Profesor može pregledavati svoj profil, evidentirati studente na nastavi, pregledavati studente po kolegiju, unositi termine za kolegije, uređivati svoje podatke, izmijeniti svoju lozinku te se odjaviti.

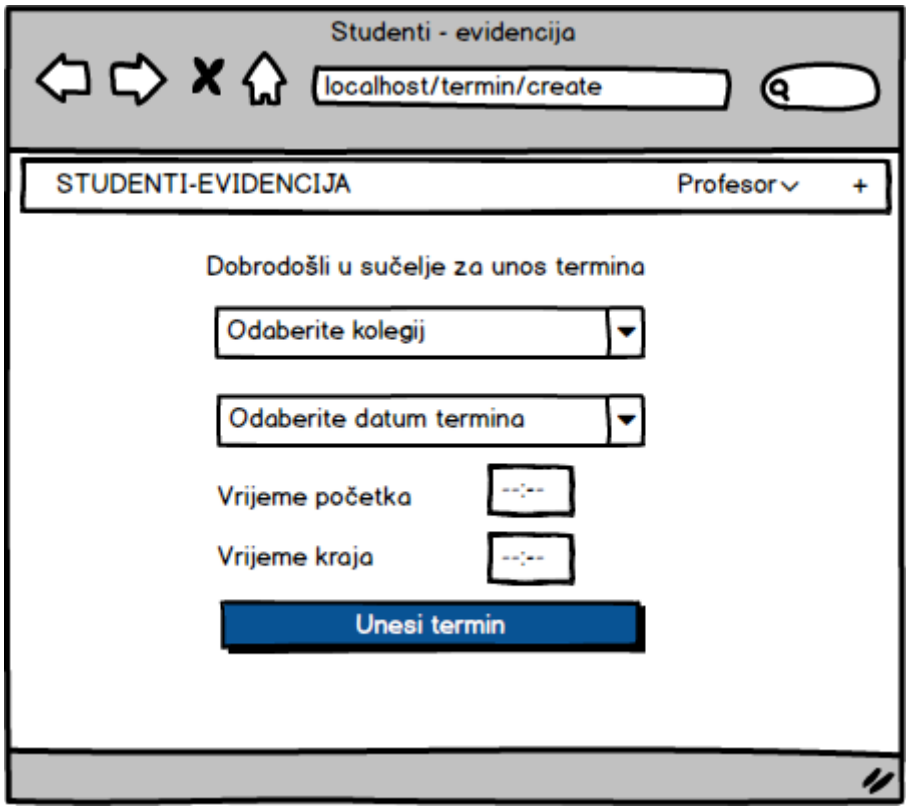
Slika 12. Prikaz početnoga profesorskog sučelja web aplikacije



Izvor: Autor

Nakon što profesor iz svog izbornika odabere dodavanje novog termina otvara se web stranica za dodavanje termina (Slika 13). Prilikom unosa termina potrebno je izabrati kolegij za koji se isti unosi. Naravno, profesor može samo unositi termine za svoje kolegije. Nakon odabira kolegija, odabire se datum termina. Kad se odabrao kolegij i datum termina potrebno je unijeti vrijeme početka termina i vrijeme kraja termina. Nakon svih odabranih i unesenih podataka pritišće se tipka „Unesi termin“ za unos termina u sustav.

Slika 13. Prikaz web stranice za unos termina



Izvor: Autor

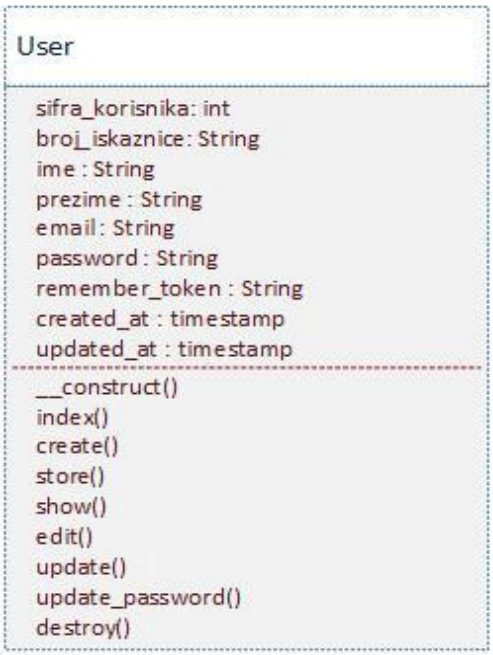
5. Kreiranje modela podataka

5.1. UML dijagram klasa

Osnovni tvorbeni elementi dijagrama su klase. One su opis skupa objekata koji dijele istu specifikaciju karakteristika (atributi i operacije), ograničenja i semantike. U mnogo slučajeva klasa je predložak iz koje je objekt sačinjen, odnosno klasa je prezentacija objekta. Za izgradnju objektno orijentiranih aplikacija klase su temelj formiranja aplikacija. Objekt je entitet iz stvarnog svijeta ili koncept (osoba, mjesto, stvar, događaj, prizor, koncept, izvještaj u sustavu). Objekti imaju svoja svojstva i za njih se kaže da „znaju stvari“ i „čine stvari“.

Kod izrade dijagrama klasa bitno je odrediti naziv klase, njezine atribute i operacije. Klasa se prikazuje pravokutnikom s nazivom na vrhu. Bitno je napomenuti da se prvo slovo klase piše velikim slovom. U sredini pravokutnika navode se atributi koje objekt ima (npr. objekt student ima atribute: ime, prezime, broj iskaznice, email,..), a u donjem dijelu slijede operacije (pod operacije unose se funkcije koje se odvijaju, npr. funkcija „*create()*“ kreira novog studenta,...). Sljedeća slika (Slika 14) prikazuje primjer jedne klase, njenih atributa i operacija.

Slika 14. Prikaz primjera klase



Izvor: Autor

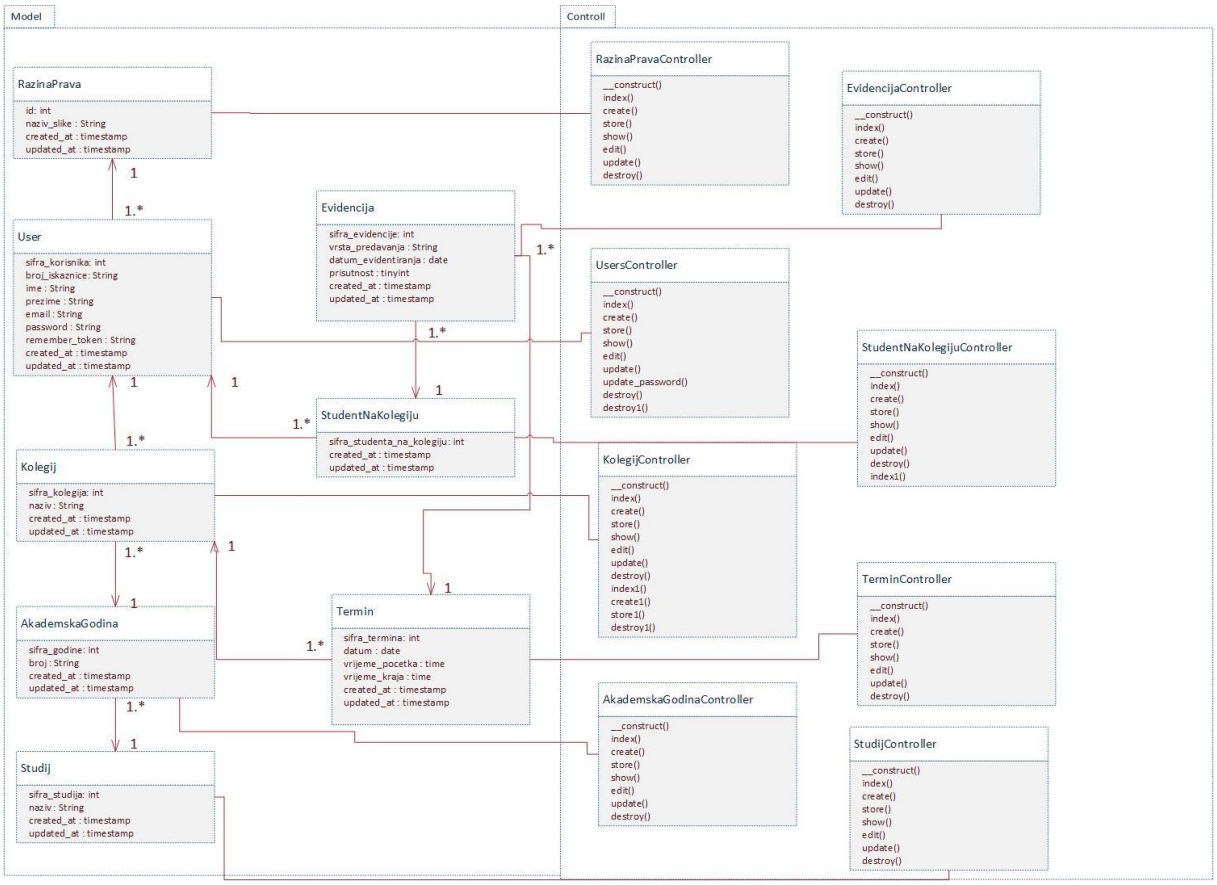
Npr. Tisuće studenata pohađa fakultet. Modelirat će se za sve studente jedna klasa u ovom slučaju „*User*“ koja prikazuje cijeli skup studenata. Za ovaj sustav izrađen je dijagram klasa te su klase podijeljene u dvije sekcije: *model* i *controll*. Klase u sekciji *model* imaju naziv klase i atribute (nemaju operacije), dok klase u sekciji *controll* imaju naziv klase i operacije. U ovom slučaju jedna klasa je podijeljena u dvije koje su međusobno povezane (npr. klasa *User* na prethodnoj slici je podijeljena u klasu „*User*“ i klasu „*UserControll*“).

U ovom sustavu postoje dvije vrste tipova klasa: domenske i kontrolne klase. Domenske klase su one klase koje sadrže jezgru aplikacijske domene. One čuvaju informacije o postojanim

aplikacijskim entitetima. Također omogućavaju spremanje i dohvaćanje atributa entiteta, kreiraju i miču entitet, te omogućuju ponašanje koje se mijenja kako se s vremenom mijenja entitet. Kontrolne klase mogu poslužiti kao kontrolori za poslove koji trebaju implementirati puteve kroz slučajeve korištenja. Njihov životni vijek je dugačak onoliko koliko je potrebno da se interakcija završi ili onoliko koliko traje korisnička sesija.

Domenske klase ovog sustava su: „RazinaPrava“, „Evidencija“, „StudentNaKolegiju“, „Kolegij“, „AkademskaGodina“, „Termin“, „Studij“ i „User“. Kontrolne klase sustava su: „RazinaPravaController“, „EvidencijaController“, „StudentNaKolegijuController“, „KolegijController“, „AkademskaGodinaController“, „TerminController“, „StudijController“, „UserController“. Domenske i kontrolne klase vidljive su na dijagramu klasa (Slika 15).

Slika 15. Prikaz dijagrama klase



Izvor: Autor

5.2. EVA model podataka

Entiteti, veze i atributi su osnovne komponente koje sačinjavaju svaki EVA model. Veze povezuje entitete, a entiteti su opisani atributima.

Entiteti su osnovni elementi o kojima se prikupljaju informacije i za koje se mogu odrediti neke karakteristike, oni su objekti baze koji dijele iste karakteristike. Primjer entiteta je bilo što za što ima smisla napraviti tablicu u kojoj bi svaki red predstavljao instancu tog entiteta, npr. u tablici studij svaki red bi predstavljao jedan studij (Stručni studij, Specijalistički studij,…). Postoje dvije vrste zavisnosti entiteta: jak nezavisan entitet koji ne ovisi o drugim entitetima već djeluje samostalno i slab nezavisni entitet koji ne postoji samostalno već ovisi o drugom entitetu. Postoji još i agregirani tip entiteta koji je specijalna vrsta slabog tipa entiteta oslabljena na dvije strane koja nema svoj primarni ključ već ga dobiva iz drugih entiteta.

Atributi su podatci pomoću kojih se opisuju entiteti. Svaki entitet ima svoj skup atributa kojima se opisuje. Razlikuju se identifikacijski i opisni atributi. Identifikacijski atributi su primarni ključevi entiteta. Oni jednoznačno određuju svaku instancu entiteta. Opisni atributi služe za detaljniji opisa instance entiteta.

Veze predstavljaju odnose među entitetima (objektima). Veza označava brojčanu ovisnost između entiteta, odnosno koliko se puta entitet s jedne strane veze pojavljuje u entitetu s druge strane veze. Kako bi se veze ostvarile, potrebno je učiniti odgovarajuću stvar s primarnim ključem entiteta koji su povezani. Postoje više vrsta veza:

- binarna 1:1 ili jedan na jedan
- ternarna 1:M ili jedan na više
- n-arna M:N ili više na više

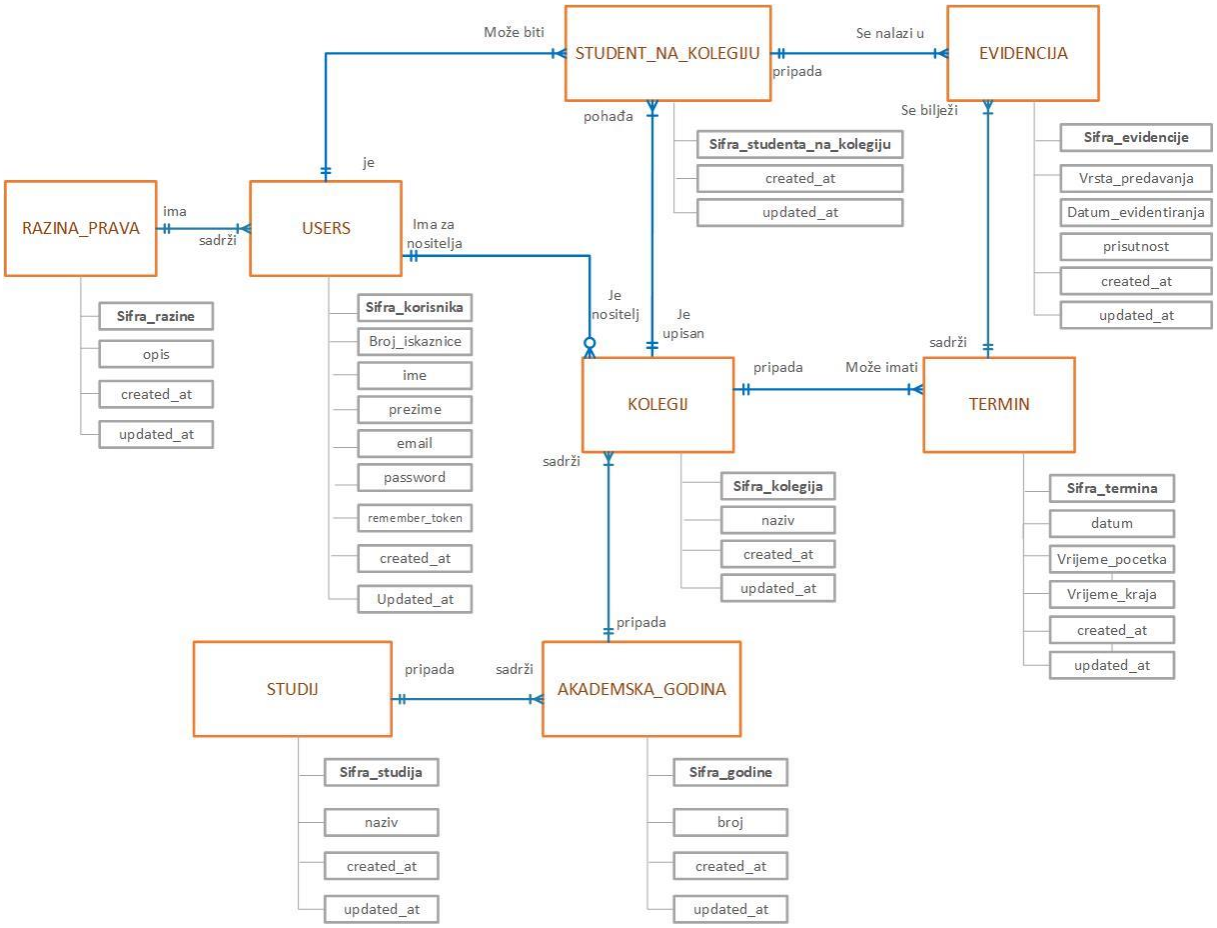
Kod veze jedan na jedan, primarni ključ jednog entiteta se uključuje u drugi entitet i naziva se vanjski ključ, jer je ključ određenog entiteta, ali se trenutno nalazi u drugome entitetu. U slučaju veze jedan na više, primarni ključ se uvijek vadi iz entiteta na strani jedan, i uvrštava kao vanjski ključ na stranu više. Kod veze više na više potrebno je izvaditi primarne ključeve oba entiteta i uvrstiti ih u novu tablicu, tablicu veze. Kad god se stvara tablica veze, moguće je u nju uvrstiti i dodatne atribute potrebne toj vezi. (Mileusnić, 2012.)

EVA model ovog sustava (Slika 16) sadržava osam entiteta s pripadajućim atributima. Svi entiteti su međusobno povezani vezama. Entitet „Razina prava“ povezan je s entitetom „*Users*“ vezom „jedan na više“ što znači da razina prava sadrži jednog ili više korisnika, odnosno suprotno, entitet „*Users*“ povezan je s entitetom „Razina prava“ vezom „jedan na jedan“ što znači da korisnik ima jednu i samo jednu razinu prava. Entitet „*Users*“ osim što je povezan s entitetom „Razina prava“, povezan je još s dva entiteta: „Student na kolegiju“ i „Kolegij“. S entitetom „Student na kolegiju“ povezan je vezom „jedan na više“, a to znači da korisnik može biti jedan ili više student na kolegiju (može pohađati više kolegija). Obratno entitet „Student na kolegiju“ povezan je s entitetom „*Users*“ vezom „jedan na jedan“, što znači da je student na kolegiju jedan i samo jedan korisnik. Također entitet „Kolegij“ povezan je s entitetom „*Users*“ vezom „jedan na jedan“ odnosno obratno vezom „nula na više“, što znači da kolegij ima za nositelja jednog i samo jednog korisnika, dok korisnik može biti nositelj više kolegija, a ujedno ne mora biti nositelj niti jednom kolegiju. Entitet „Student na kolegiju“ osim što je povezan s entitetom „*Users*“ povezan je još s dva entiteta: „Evidencija“ i „Kolegij“. S entitetom „Kolegij“ povezan je vezom „jedan na jedan“ i obratno entitet „Kolegij“ povezan je s entitetom „Student na kolegiju“ vezom „jedan na više“. Svaki student na kolegiju je upisan na jedan i samo jedan kolegij, a kolegij pohađa jedan ili više studenata na kolegiju. Drugi entitet „Evidencija“ povezan je s entitetom „Student na kolegiju“ vezom „jedan na jedan“ i obratno vezom „jedan na više“, što znači da evidencija pripada jednom i samo jednom studentu, dok se student na kolegiju nalazi u jednoj ili više evidencija. Entitet „Evidencija“ još je povezana s još jednim entitetom – „Termin“ također vezom „jedan na jedan“ i obratno vezom „jedan na više“. Evidencija sadrži jedan i samo jedan termin, a termin se bilježi u jednoj ili više evidencija

Entitet „Kolegij“ u ovom EVA modelu podataka najviše je povezan čak s četiri entiteta. Osim što je povezan entitetima „*Users*“ i „Student na kolegiju“ koji su prethodno opisani, povezan je još s entitetima „Termin“ i „Akademska godina“. Tako je s entitetom „Termin“ povezan vezom „jedan na više“ odnosno obratno „jedan na jedan“, što znači da kolegij može imati jedan ili više termina, dok termin pripada jednom i samo jednom kolegiju. S entitetom „Akademska godina“ povezan je vezom „jedan na jedan“, tj. obratno vezom „jedan na više“. To bi značilo da kolegij pripada jednoj i samo jednoj akademskoj godini, a akademska godina sadrži jedan ili više kolegija. Posljednja veza je između entiteta „Akademska godina“ i „Studij“. Entitet „Akademska godina“ povezan je s entitetom „Studij“ vezom „jedan na jedan“, odnosno

obratno vezom „jedan na više“. Tako akademska godina pripada jednom i samo jednom studiju, dok studij sadrži jednu ili više akademskih godina.

Slika 16. Prikaz EVA modela



Izvor: Autor

5.3. Relacijski model podataka

Nakon izrade EVA modela, pravi se relacijski model poštujući pravila. Relacijski model je konceptualni temelj relacijske baze podataka. Predložen od E.F. Codd-a 1969. Relacijski model je metoda strukturiranja podataka koristeći relacije koje su nalik na mrežaste matematičke strukture koje se sastoje od redova i stupaca. Codd je predložio relacijski model za IBM (*International Business Machines*), ali nije imao pojma od kolike važnosti će njegov rad biti u stvaranju temelja relacijskih baza podataka.

U relacijskom modelu svi podaci moraju biti pohranjeni u relacije (tablice), a svaka relacija se sastoji od redaka i stupaca. Svaka relacija treba imati zaglavlje i tijelo. Zaglavlje je lista stupaca u relaciji. Tijelo čine podaci koji u biti popunjavaju te relacije organizirane u redove. Izvodi se vrijednost koja govori da spajanje jednog reda i jednog stupca čini jednoznačnu, unikatnu vrijednost, a ta vrijednost se naziva „torka".

Druga vrlo glavna karakteristika relacijskog modela jest korištenje ključeva. To su posebno određeni stupci u relaciji koji se koriste za slaganje ili povezivanje podataka s drugim relacijama. Jedan od najvažnijih ključeva jest primarni ključ koji se koristi za jednoznačno identificiranje svakog retka podataka. Kako bi se upiti nad podacima vršili na lakši način većina relacijskih baza podataka je otišla korak dalje te se podaci fizički slažu po primarnom ključu. Vanjski ključevi povezuju podatke u jednoj relaciji s primarnim ključem druge relacije.

Osim definiranja načina na koji će se podaci strukturirati kako je diskutirano gore, relacijski model također podliježe skupu pravila kako bi se ojačao integritet podataka poznatiji kao ograničenje referencijalnog integriteta. Ograničenja referencijalnog integriteta rade na konceptu vanjskih ključeva. Vanjski ključ ključni je atribut relacije koji se može odnositi i na neku drugu relaciju. Ograničenja referencijalnog integriteta navode da ako relacija odgovara ključnom atributu iste ili različite relacije, tada taj ključni atribut mora postojati. (Janssen, 2016)

Relacijski model ovog sustava (Slika 17) sastoji se od osam tablica. Sve tablice su međusobno povezane preko vanjskih ključeva. Tablica „users" povezana je više tablica. S tablicom „razina_prava" povezana je preko vanjskog ključa „razina_prava" koji je primarni ključ u tablici „razina prava" (sifra_razine). S tablicom „student_na_kolegiju" povezana je preko vanjskog ključa „sifra_korisnika" koji je primarni ključ u tablici „users"

(sifra_korisnika), te zatim povezana je s tablicom „kolegij“ preko vanjskog ključa „sifra_profesora“ koji je u tablici „users“ primarni ključ (sifra_korisnika). Vanjski ključ u nekoj tablici je primarni ključ u nekoj drugoj tablici, te nazivi ključeva se mogu razlikovat kao u ovom slučaju gdje je vanjski ključ „sifra_profesora“ u tablici „kolegij“ primarni ključ u tablici „users“ pod nazivom „sifra_korisnika“.

Tablica „student_na_kolegiju“ osim što je povezana s tablicom „users“ povezana je s još dvije tablice („evidencija“ i „kolegij“). S tablicom „evidencija“ povezana je preko vanjskog ključa „sifra_studenta_na_kolegiju“ koji je ujedno u tablici „student_na_kolegiju“ primarni ključ (sifra_studenta_na_kolegiju), te s tablicom „kolegij“ povezana je preko vanjskog ključa „sifra_kolegija“ koji je u tablici „kolegij“ primarni ključ (sifra_kolegija). Tablica „evidencija“ osim što je povezana s tablicom „student_na_kolegiju“ povezana je također s tablicom „termin“ preko vanjskog ključa „sifra_termina“ koji je primarni ključ u tablici „termin“ (sifra_termina).

Najviše međusobnih veza ima tablica „kolegij“ koja je povezana s četiri tablice. Povezana je s tablicom „termin“ preko vanjskog ključa „sifra_kolegija“ koji je primarni ključ u tablici „kolegij“ (sifra_kolegija), a s tablicom „akademska_godina“ povezana je preko vanjskog ključa „sifra_godine“ koji je primaran ključ u tablici „akademska_godina“ (sifra_godine). Dvije veze tablice „kolegij“ su opisane prethodno. Na slici se može vidjeti da tablica „kolegij“ vuče dva vanjska ključa (tablice „users“ i „akademska_godina“), te da od tablice „kolegij“ druge tablice („student_na_kolegiju“ i “termin“) vuku njezin primarni ključ kao što je opisano prethodno međusobnim vezama . Zadnji opis međusobnih veza su između tablica „akademska_godina“ i „studij“. Povezane su preko vanjskog ključa „sifra_studija“ koji je u tablici „studij“ primarni ključ (sifra_studija).

Slika 17. Prikaz relacijskog modela

Izvor: Autor

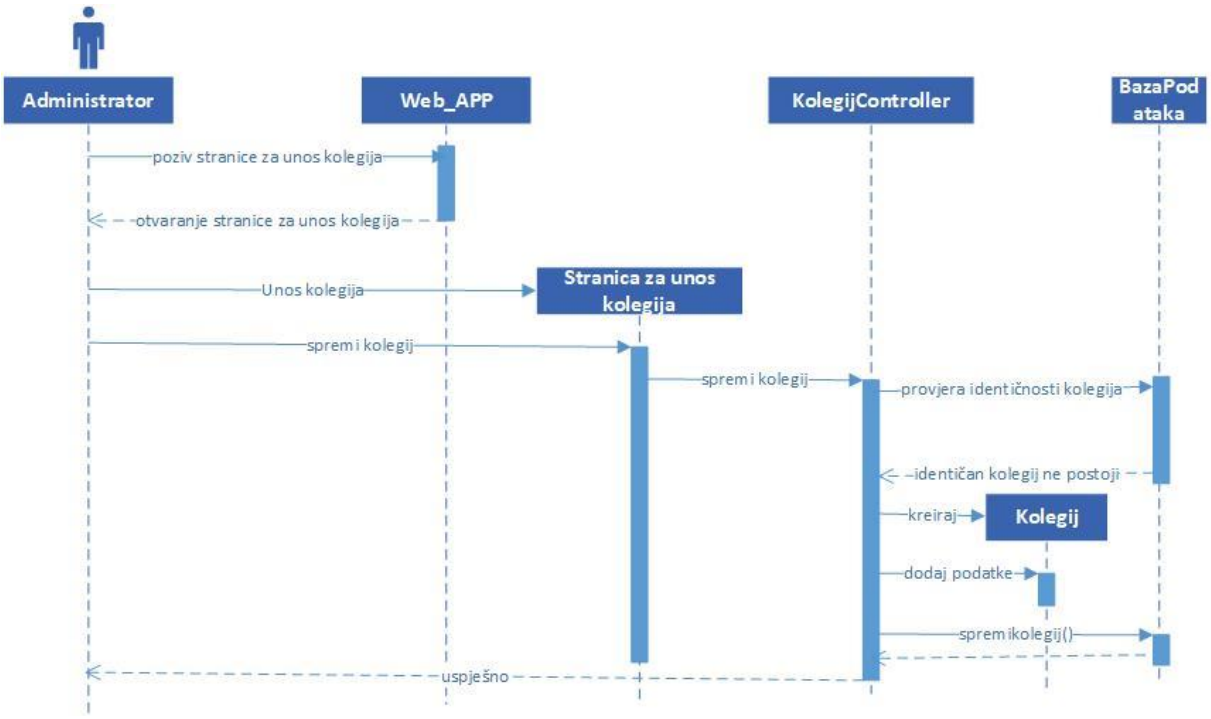
6. Objektno orijentirani dizajn

U ovom poglavlju prikazat će se sekvencijalni dijagrami za slučajeve korištenja (Poglavlje 3.). **Sekvencijalni dijagram** prvenstveno opisuje kako će sustav napraviti određene zadatke, za razliku od dijagrama slučajeva korištenja koji opisuje što sustav treba raditi. Modeliraju tok logike unutar sustava u vizualnom smislu. Naglasak je na vremenskom redoslijedu kojim se odvija interakcija među dijelovima sustava. Sekvencijalni dijagram pokazuje kronološki redoslijed događaja i operacija. Sekvencijalni dijagram je vremenski orijentiran, tj. prati linearan tijek poruka između objekata, a vrijeme teče od početka dijagrama prema dolje. U dinamičkom modeliranju najkorišteniji je dijagram. Zajedno s dijagramima klasa i fizičkim modelom podataka najvažniji su modeli za dizajniranje modernih poslovnih aplikacija. Pomoću njih definiraju se objekti (sudionici) i poruke među njima (sinkrona, asinkrona, povratna poruka, poruka kreiranja, poruka brisanja). Ime su dobili po sekvenciji, druga sekvencija slijedi iz prve.

6.1. Sekvencijalni dijagram – unos, izmjena i brisanje kolegija

Najprije će se prikazat sekvencijalni dijagram za unos kolegija. Proces se odvija na način da administrator pristupa web aplikaciji i poziva stranicu za unos kolegija. Nakon otvaranja stranice, administrator unosi podatke o kolegiju, te ih zatim sprema. Proces spremanja podataka o kolegiju putuje preko klase „*KolegijController*“ koja provjerava identičnost kolegija (da li kolegij već postoji za određen studij i godinu studija) u bazi podataka. Nakon odgovora baze podataka da identičan kolegij ne postoji, klasa „*KolegijController*“ kreira novi objekt tipa „*Kolegij*“ i u njega unosi podatke. Zatim klasa „*KolegijController*“ izvršava funkciju „*spremikolegij()*“ koja sprema unesene podatke o kolegiju u bazu podataka i obavještava administratora o uspješnom spremanju kolegija (Slika 18).

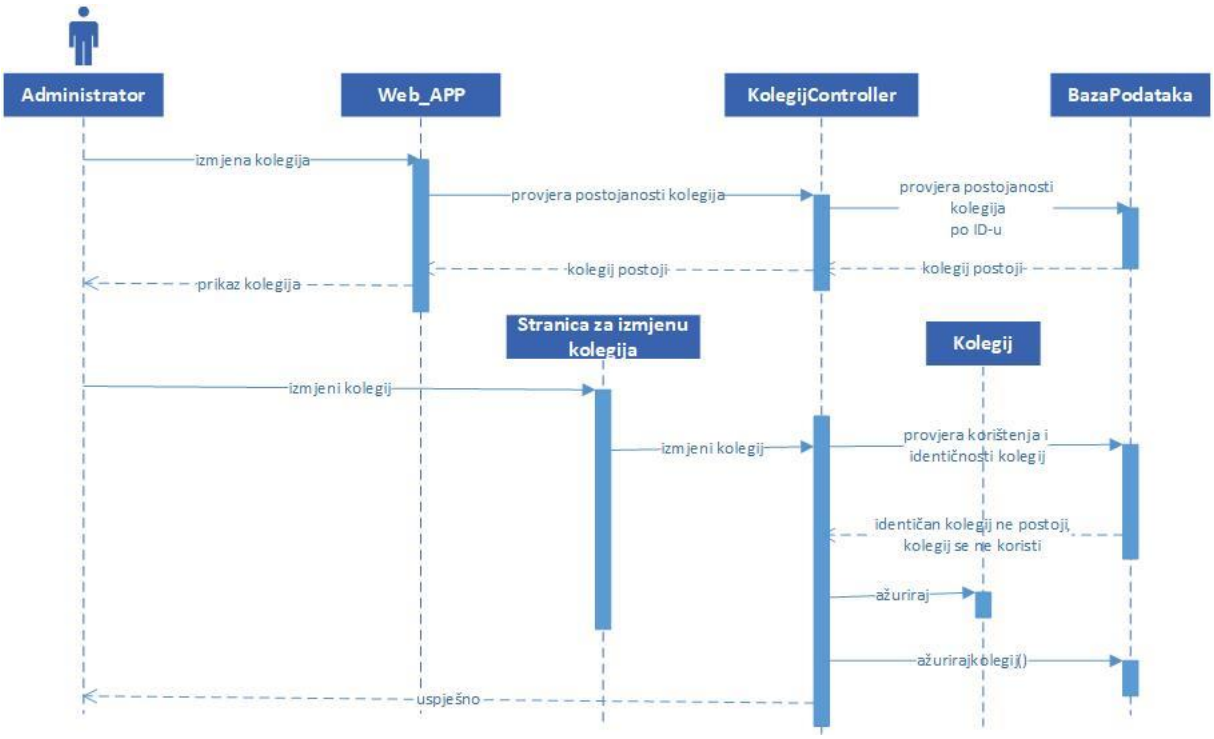
Slika 18. Prikaz sekvencijalnog dijagrama za unos kolegija



Izvor: Autor

Sljedeća slika (Slika 19) prikazuje sekvencijalni dijagram za izmjenu kolegija. Proces se odvija na način da administrator pristupa web aplikaciji, te zatim poziva stranicu za izmjenu kolegija. Proces za poziv stranice za izmjenu kolegija putuje preko klase „*KolegijController*“ koja provjerava postojanje kolegija po šifri u bazi podataka. Baza podataka vraća poruku klasi „*KolegijController*“ da kolegij s tom šifrom postoji, te klasa „*KolegijController*“ šalje poruku odnosno otvara stranicu za izmjenu odabranog kolegija. Nakon što se stranica otvorila administrator izmjenjuje podatke o kolegiju. Nakon izmjene podataka sprema promjene. Proces spremanja promijenjenih podataka putuje preko klase „*KolegijController*“ koja provjerava u bazi podataka provjeru korištenja i identičnosti kolegija (koristi li se kolegij negdje i postoji li već kolegij s takvim podatcima). Baza podataka odgovara da se kolegij ne koristi i da ne postoji takav kolegij s tim podatcima. Klasa „*KolegijController*“ preko klase modela „*Kolegij*„ ažurira postojeći objekt „*Kolegij*“ i u njega dodaje nove podatke. Nakon toga klasa „*KolegijController*“ izvršava funkciju „*ažurirajkolegij()*“ koja sprema izmijenjene podatke o kolegiju u bazu podataka, te obavještava administratora o uspješnoj izmjeni kolegija.

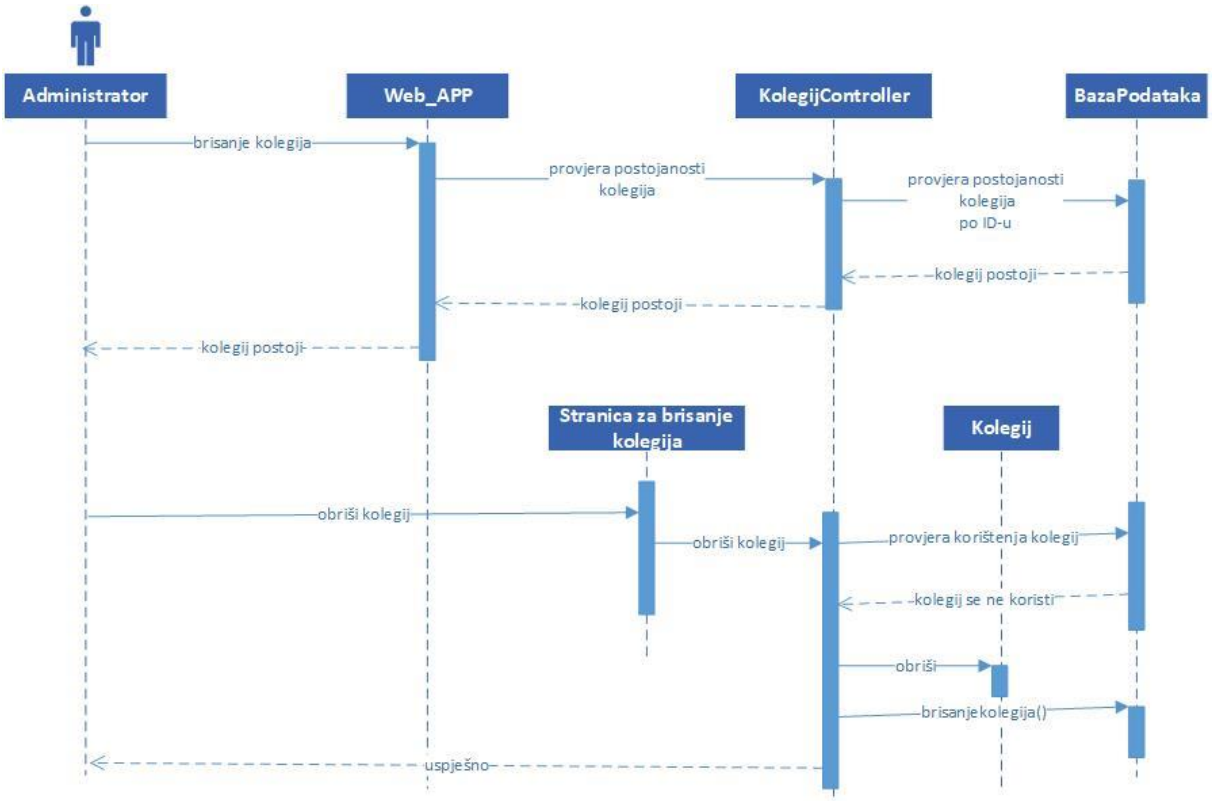
Slika 19. Prikaz sekvencijalnog dijagrama za izmjenu kolegija



Izvor: Autor

Nakon opisa i grafičkog prikaza za unos i izmjenu kolegija, opisat će se i grafički prikazat brisanje kolegija. Proces brisanja kolegija odvija se na način da administrator pristupa web aplikaciji, te poziva stranicu za brisanje određenog kolegija. Proces izabranog kolegija putuje preko klase „*KolegijController*“ koja provjerava postojanje kolegija po šifri u bazi podataka. Baza podataka odgovara klasi „*KolegijController*“ da odabrani kolegij s tom šifrom postoji u sustavu. Klasa „*KolegijController*“ otvara stranicu za brisanje kolegija. Administrator briše kolegij. Proces brisanja kolegija putuje preko klase „*KolegijController*“ koja provjerava korištenje kolegija u bazi podataka (da li su studenti pohađaju taj kolegij). Baza podataka vraća poruku da se kolegij ne koristi, te klasa „*KolegijController*“ preko klase modela „*Kolegij*“ briše postojeći objekt „*Kolegij*“. Zatim klasa „*KolegijController*“ izvršava funkciju „*brisanjekolegija()*“ koja briše podatke o kolegiju iz baze podataka, te obavještava administratora o uspješnom brisanju kolegija (Slika 20).

Slika 20. Prikaz sekvencijalnog dijagrama za brisanje kolegija

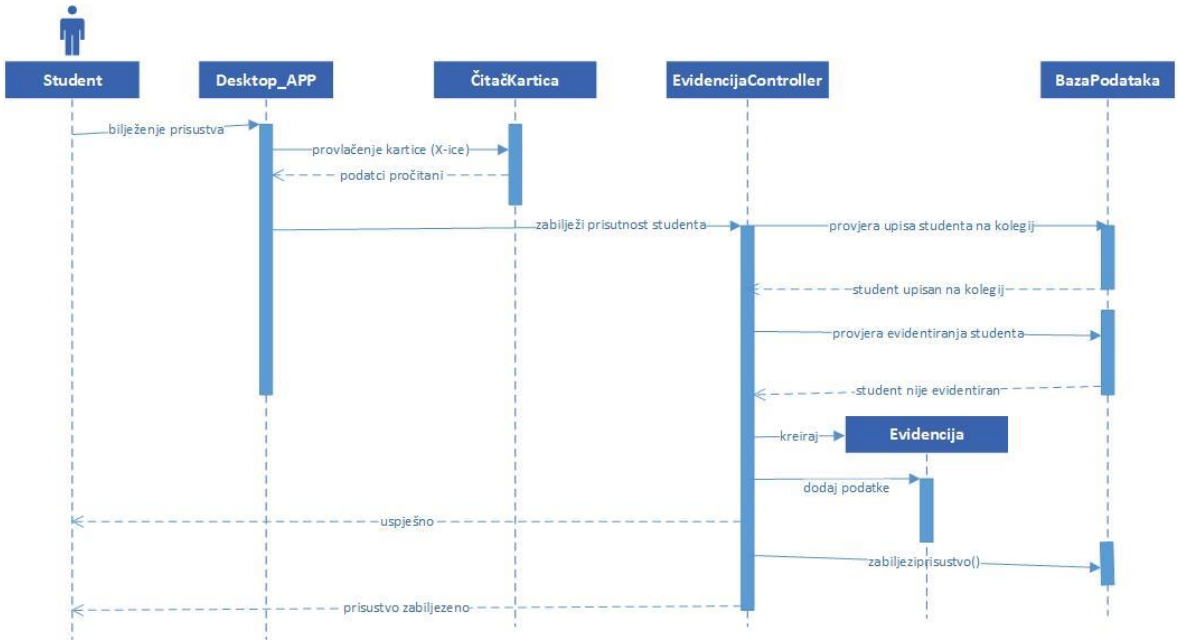


Izvor: Autor

6.2. Sekvencijalni dijagram – evidentiranje studenata preko čitača kartica

Proces evidentiranja studenta preko čitača kartica odvija se na način da student pristupa desktop aplikaciji gdje bilježi prisustvo tako da provlači svoju karticu (X-icu) kroz čitač kartica. Nakon provlačenja kartice čitač kartica obrađuje podatke koje je pročitao s magnetske trake kartice te ih šalje desktop aplikaciji. Nakon što je aplikacija dobila informacije o studentu automatski se pokreće funkcija za evidentiranje studenta. Taj proces odvija se na način da klasa „*EvidencijaController*“ provjera u bazi podataka da li je student upisan na kolegij za koji prijavljuje svoju prisutnost. Baza podataka šalje povratnu informaciju da je student upisan na kolegij. Klasa „*EvidencijaController*“ zatim provjerava evidenciju studenta tj. da li je student evidentiran već za taj termin na kojem prijavljuje svoju prisutnost (putem desktop ili web aplikacije). Baza podataka šalje povratnu informaciju da student nije evidentiran. Zatim klasa „*EvidencijaController*“ kreira novi objekt tipa „*Evidencija*“ i u njega unosi podatke. Nakon kreiranja objekta, klasa „*EvidencijaController*“ izvršava funkciju „*zabiljeziprisustvo()*“ koja bilježi prisutnost studenta na termin nastave u bazu podataka i obavještava studenta o uspješnoj evidenciji (Slika 21).

Slika 21. Prikaz sekvencijalnog dijagrama za evidentiranje studenata preko čitača kartica



Izvor: Autor

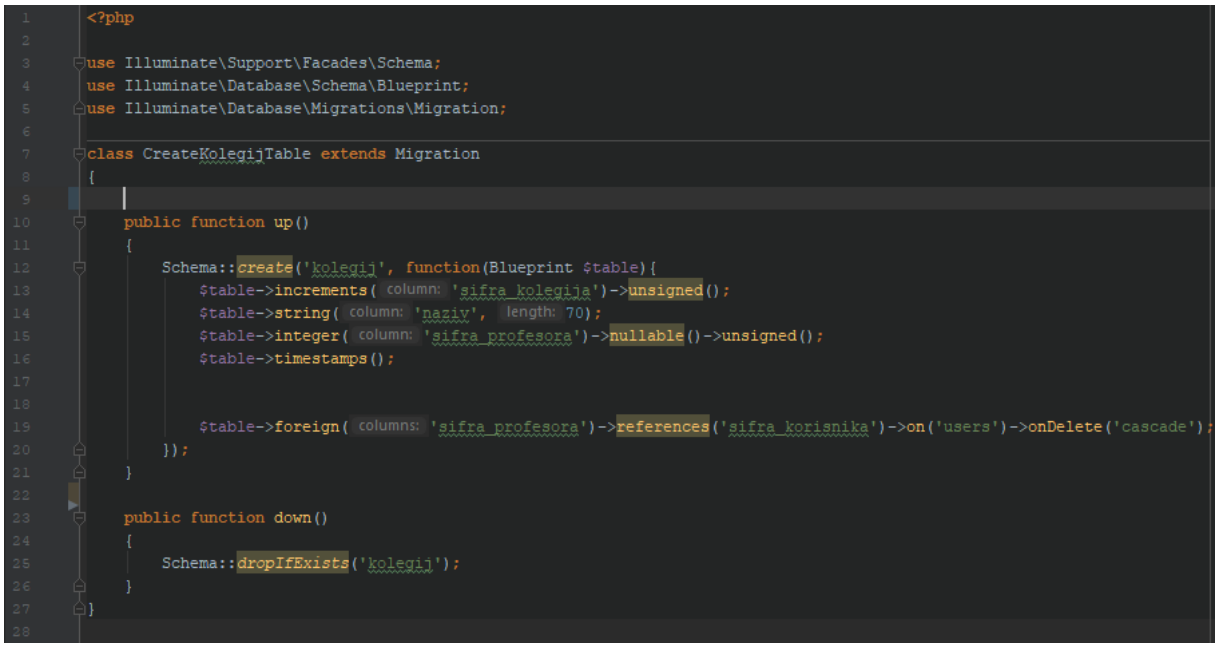
7. Implementacija sustava u Laravel razvojnom okruženju

Web aplikacija ovog sustava je izrađena u Laravelu. Korištena verzija Laravela za izradu aplikacije je 5.4. Aplikacija je izrađena prema MVC pristupu. Kod izgradnje MVC pristupa započinje se s modelima, zatim se kreiraju pregledi, te posljednji kontrolori. Laravel dolazi sa ugrađenim alatima kao što je opisano u poglavlju 2.2.1.

Laravelov alat „Artisan“ ima dosta funkcija. Jedna od njih je kreiranje migracija koje se koriste prilikom izrade baze podataka. Migracije se koriste kao oblik kontroliranja baze podataka. Za stvaranje tablica koriste se migracije tako da se u naredbenom retku ukuca naredba: „php artisan make:migration create_nazivTablice_table --create nazivTablice“. Ta naredba će stvoriti datoteku naziva ”year_month_day_hour_minute_second_create _nazivTablice_table” u direktoriju *database/migrations* u kojoj je definirana tablica „nazivTablice“. U nazivu migracije Laravel dodaje ispred imena migracije godinu, mjesec, dan, sat, minute i sekunde kako bi pratio vrijeme stvaranja pojedinih migracija. To je vrlo važno jer prilikom stvaranja treba voditi računa da se najprije rade migracije koje mogu postojati samostalno tj. nemaju nikakve vanjske ključeve. U suprotnom Laravel će javiti grešku i migracije neće biti pretvorene u tablice u bazi podataka. Kreirana migracija je klasa koja se sastoji od dvije funkcije: „up()“ i „down()“. Metoda „up()“ služi za dodavanje novih tablica, stupaca i indeksa u bazu podataka. Metoda „down()“ služi za poništavanje svih akcija koje je provela metoda „up()“. Nakon kreiranja svih migracija izvodi se naredba „php artisan migrate“ kako bi se migracije pretvorile u tablice u bazi podataka. U bilo kojem trenutku tablica se može mijenjati (dodavati, brisati, mijenjati stupce). Ukoliko je tablica mijenjana nakon njenog stvaranja i unošenja podataka, potrebno je ostatak baze podataka prilagoditi promjenama i ponovo migrirati.

Na sljedećem primjeru grafički će se prikazati programski kod migracije kod kreiranja kolegija (Slika 22). Tablica se stvara koristeći metodu *create* Laravel fasade *Shema* koja prima argumente: naziv tablice, funkcija koja prima *Blueprint* objekt. Preko objekta se popunjava tablica sa potrebnim podatcima. U ovom primjeru u tablici „kolegij“ dodani su stupci: šifra kolegija, naziv, šifra profesora, datum kreiranja, te je napravljen vanjski ključ koji je u tablici *users* primarni ključ.

Slika 22. Prikaz programskog koda migracije u Laravelu



```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateKolegijTable extends Migration
{

    public function up()
    {
        Schema::create('kolegij', function(Blueprint $table){
            $table->increments( column: 'sifra_kolegija')->unsigned();
            $table->string( column: 'naziv', length: 70);
            $table->integer( column: 'sifra_profesora')->nullable()->unsigned();
            $table->timestamps();


            $table->foreign( columns: 'sifra_profesora')->references('sifra_korisnika')->on('users')->onDelete('cascade');
        });
    }

    public function down()
    {
        Schema::dropIfExists('kolegij');
    }
}
```

Izvor: Autor

Nakon kreiranja baze podataka, započeto je kreiranje web aplikacije. U nastavku prikazat će se MVC pristup kod registracije korisnika u sustav. Najprije se izrađuje model korisnika. Model u Laravelu je moguće kreirati preko naredbenog retka i naredbe „php artisan make:model NazivModela“ ili ručno u mapi „app“. Nakon kreiranja modela potrebno je unijeti potrebne podatke u njega. Laravel primarni ključ svake tablice po zadanim postavkama prepoznaje po nazivu „id“. Zato treba ukoliko se primarni ključ tablice tako ne zove, postaviti primarni ključ u varijablu „primaryKey“ zaštićenog tipa (*protected*) onako kako se zove. U ovom slučaju je to šifra korisnika. Nakon postavljanja primarnog ključa tablice, u varijablu *fillable* koja je zaštićenog tipa dodaju se imena stupaca u tablici. Zadnja varijabla u modelu je *hidden* koja štiti podatke tj. čini ih skrivenima npr. od nekih napada na sustav. U ovom slučaju to su korisnička lozinka, *token* korisnika, broj iskaznice korisnika i ovlast korisnika (Slika 23).

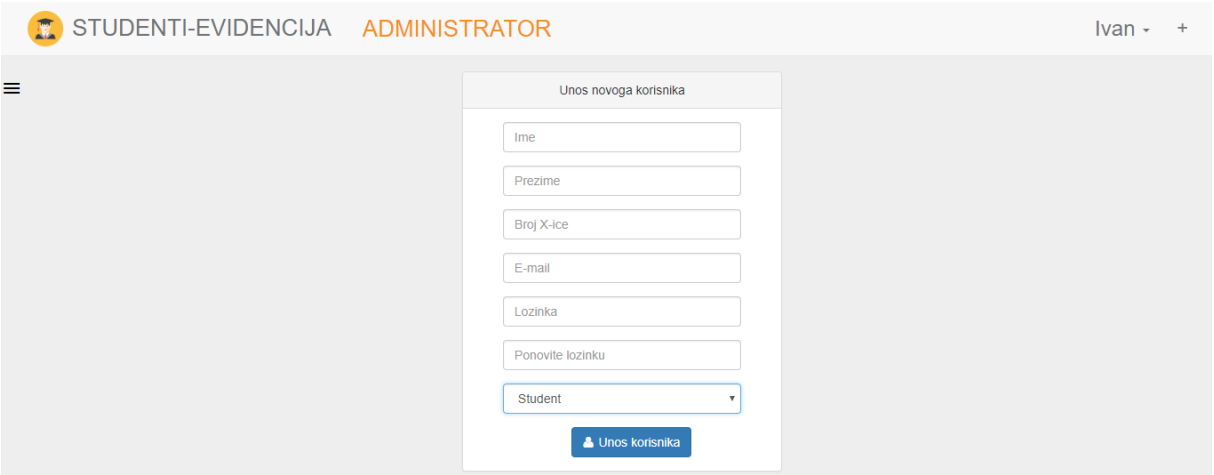
Slika 23. Prikaz programskog koda klase modela "User"

```php
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Session;

class User extends Authenticatable
{
    use Notifiable;
    protected $primaryKey = 'sifra_korisnika';

    protected $fillable = [
        'ime', 'prezime', 'password','email','razina_prava','broj_iskaznice','sifra_korisnika',
    ];


    protected $hidden = [
        'lozinka', 'remember_token','broj_iskaznice','razina_prava',
    ];


}
```

Izvor: Autor

Iza kreiranja klase modela kreiraju se pregledi. Za primjer pokazati će se pregled za unos korisnika (registraciju) u bazu podataka. Korisnik se unosi na način da profesor ili administrator popunjavaju osnovne podatke za njega. Profesor može unositi samo studente, a administrator ima mogućnost unositi osim studenata, profesore i druge administratore. Kad su podatci popunjeni pritišće se gumb „Unos korisnika“ (Slika 24).

Slika 24. Prikaz izgleda sučelja za unos korisnika na web aplikaciji



Izvor: Autor

Nakon pritiska gumba „Unos korisnika“ sustav treba unijeti istog u bazu podataka. Za to je potreban upravljački dio – kontrolor. Kontrolor se kao i model može kreirati uz pomoć naredbenog retka i naredbe „php artisan make:controller NazivControllera“ ili ručno u mapi „app/Http/Controllers“. Sve metode koje obrađuju zahtjeve se definiraju unutar kontrolora. Za spremanje korisnika koristi se funkcija *store* koja se prilikom kreiranja kontrolora već nalazi u klasi. Funkcija najprije provjerava validaciju podataka, odnosno one podatke koje je profesor ili administrator unio u pregledu za unos korisnika. Ukoliko podatci nisu dobro uneseni ili se poklapaju već s nekim podatcima iz baze podataka (*e-mail*) sustav će javiti točno koja je greška. Ako je validacija podataka uspješna, funkcija u objekt tipa *User* dodaje podatke korisnika. Zatim se ti podatci spremaju u bazu podataka. Ukoliko su podatci uspješno spremljeni, sustav će javiti profesoru ili administratoru poruku o uspješnom spremanju korisnika. U suprotnom sustav će javiti grešku da korisnik nije unesen.

Slika 25. Prikaz programskog koda klase kontrolora " UsersController "

```
public function store(Request $request)
{
    $this->validate($request,[
        'ime' => 'required|string|max:255',
        'prezime' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6|confirmed',
        'broj_iskaznice' => 'required|numeric|unique:users',
        'razina_prava' =>'required|not_in:0'

    ],[...]);

        $korisnik = new User(array(
            'ime' => $request->get( key: 'ime'),
            'prezime' =>$request->get( key: 'prezime'),
            'email' => $request->get( key: 'email'),
            'password' => bcrypt($request->get( key: 'password')),
            'broj_iskaznice' => $request->get( key: 'broj_iskaznice'),
            'razina_prava' => $request->get( key: 'razina_prava'),
        ));

            if($korisnik->save()){

            Session::flash('flash_message', 'Uspješan unos korisnika: "'.$korisnik->ime.' '.$korisnik->prezime.'" ');
            return redirect()->back();

            }else
                Session::flash('flash_message1', 'Korisnik nije unesen!');
                return redirect()->back();
}
```

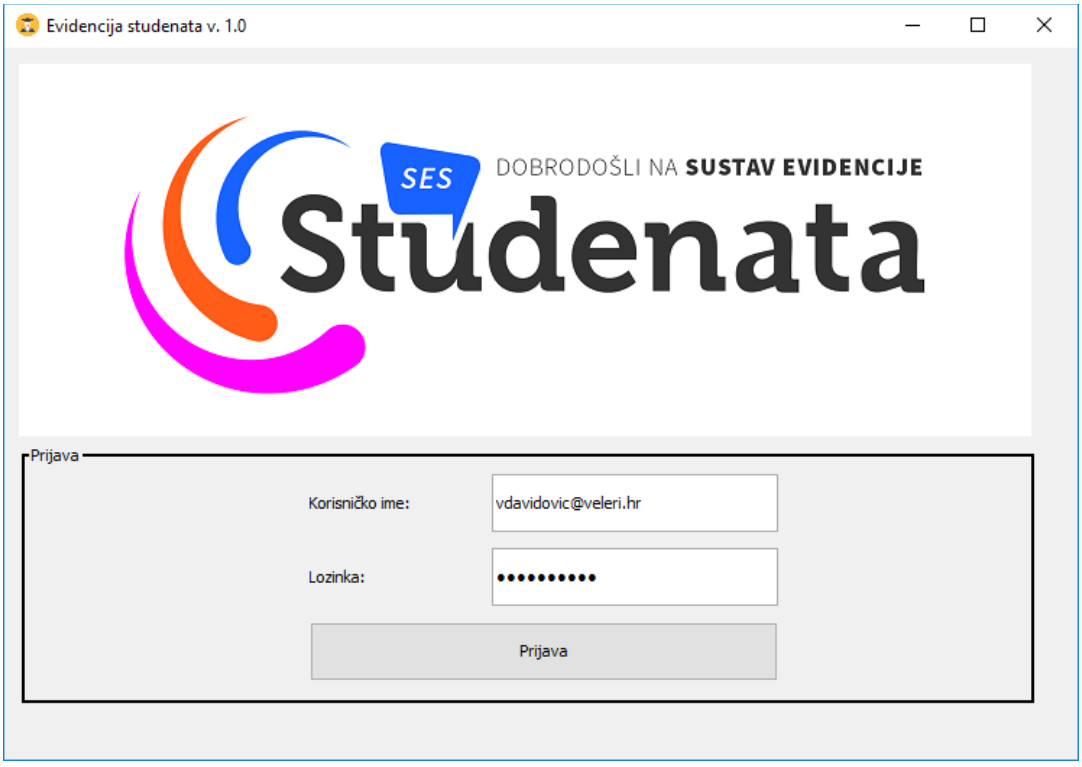
Izvor: Autor

8. Prikaz uporabe programskog rješenja

U ovom poglavlju detaljno će se obraditi evidencija studenata preko studentskih kartica (X-ica) koristeći desktop aplikaciju i čitač kartica.

Prije svega treba provjeriti internetsku vezu na računalu kako bi aplikacija mogla komunicirati s bazom podataka. Nakon provjere pokreće se desktop aplikacija. Aplikacija otvara početno sučelje koje zahtjeva korisničke podatke, odnosno korisničko ime i lozinku (Slika 26).

Slika 26. Prikaz početne stranice produkcijske desktop aplikacije

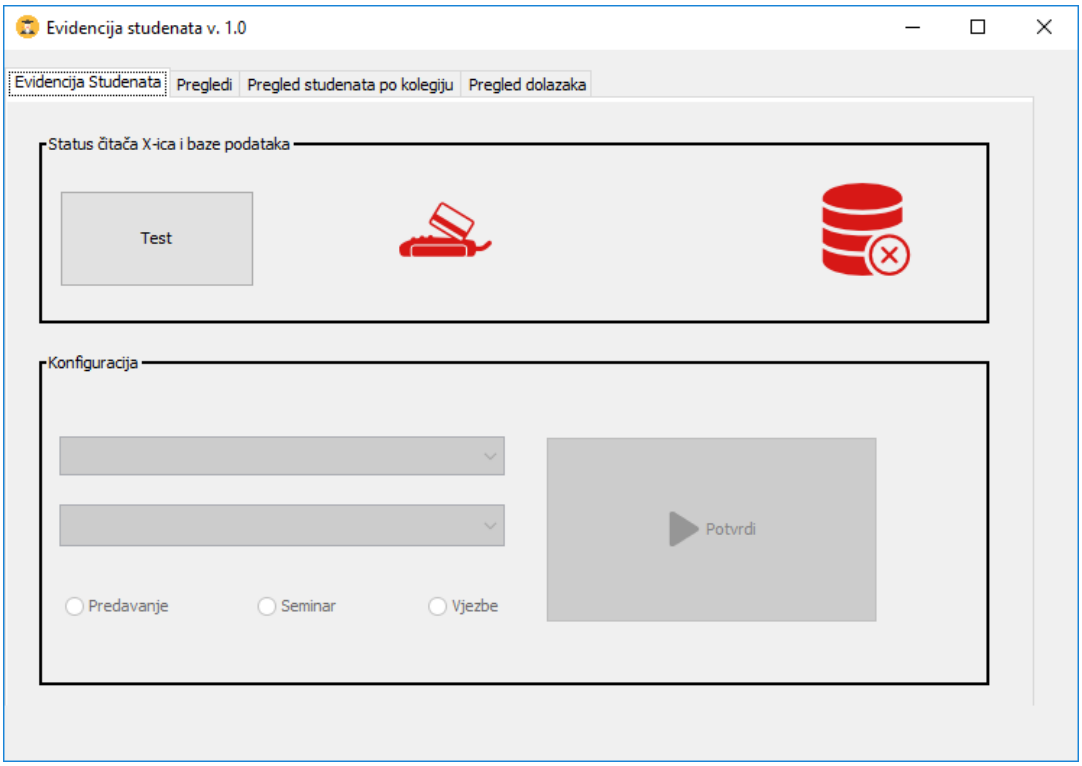


Izvor: Autor

Nakon unesenih korisničkih podataka, korisnik pritišće gumb „Prijava“ kako bi sustav provjerio validaciju podataka, te korisniku otvorio određeno sučelje. U ovom slučaju profesor se prijavljuje u sustav i sustav otvara profesorsko sučelje. Profesor ima više izbornika. Prvi

izbornik koji se otvori prilikom prijave je izbornik „Evidencija Studenata“. U tom izborniku postoje dvije sekcije. U prvoj sekciji radi se test aplikacije s čitačem kartica i bazom podataka, dok u drugoj odabire se konfiguracija za evidenciju studenata. Druga sekcija je isključena i profesor ne može odabrati podatke za početak evidentiranja studenata dok prva sekcija ne bude izvršena. U prvoj sekciji prije testiranja aplikacije s čitačem kartica i bazom podataka potrebno je spojiti čitač kartica na računalo preko USB priključka. Nakon spajanja čitača, sustav u ovom slučaju „*Windows*“ drivere instalira automatski za taj čitač. Čitač je spreman za rad ako svijetli na njemu zelena lampica. Nakon provjere internetske veze i spajanja čitača kartica pritišće se gumb „Test“ gdje sustav provjerava konekciju. Ukoliko čitač ili baza podataka ne komuniciraju s aplikacijom sustav će javiti poruku s greškom koja veze nije ostvarena. Po početnim postavkama ikone čitača i baze podataka su crvene boje što označava da aplikacija nije spojena s njima (Slika 27). U suprotnom ukoliko je konekcija aplikacije uspješna s čitačem i bazom podataka sustav treba uključiti drugu sekciju i ikone promijeniti u zelenu boju.

Slika 27. Prikaz profesorskog sučelja produkcijske desktop aplikacije 1



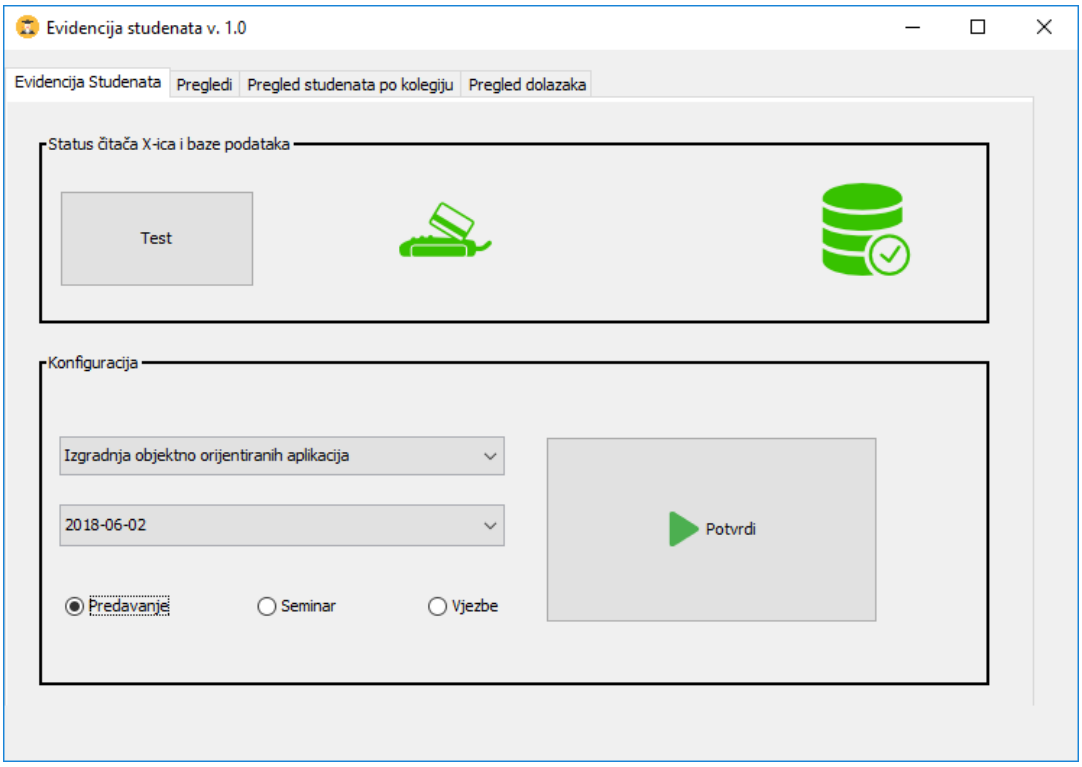
Izvor: Autor

Nakon uspješne provjere aplikacije s čitačem i bazom podataka, profesor može prijeći na drugu sekciju u kojoj će odabrati podatke potrebne za bilježenje studenata. Prvi podatak koji odabire je kolegij. Profesor može samo odabrati svoje kolegije iz liste. Nakon odabira kolegija automatski se popunjava lista termina za taj kolegij. Termin profesor mora unijeti u sustav prije evidentiranja studenata putem desktop ili web aplikacije.

Profesor može odabrati bilo koji termin iz liste termina, ali sustav mu neće dozvoliti evidentiranje studenata na bilo koji termin, nego samo na onaj termin koji je taj određeni dan. Također prilikom unosa termina postoji i unos vremena početka i vremena kraja termina. Ako je profesor ispravno odabrao termin za određeni dan, sustav mu ne mora dozvoliti daljnje radnje za evidentiranje studenata ukoliko profesor ne vodi evidentiranje u vremenskom razdoblju tog termina. Drugim riječima, ako profesor evidentira studente na dan 02.06.2018. u 15:00 sati, sustav će mu dozvoliti evidentiranje ako je prethodno uneseni termin na taj dan i ako su sati u vremenskom razdoblju vremena početka i vremena kraja (npr. vrijeme početka je 14:45 sati, a vrijeme kraja 18:00 sati, sustav će dozvoliti evidentiranje studenata).

Nakon odabira kolegija i termina ostaje odabrati još vrstu predavanja. Tri opcije su odabira vrste predavanja: predavanja, vježbe i seminar. Profesor izabire jedno od toga ovisno o planu i programu nastave. Nakon odabira kolegija, termina i vrste predavanja pritišće se gumb „Potvrdi“ (Slika 28).

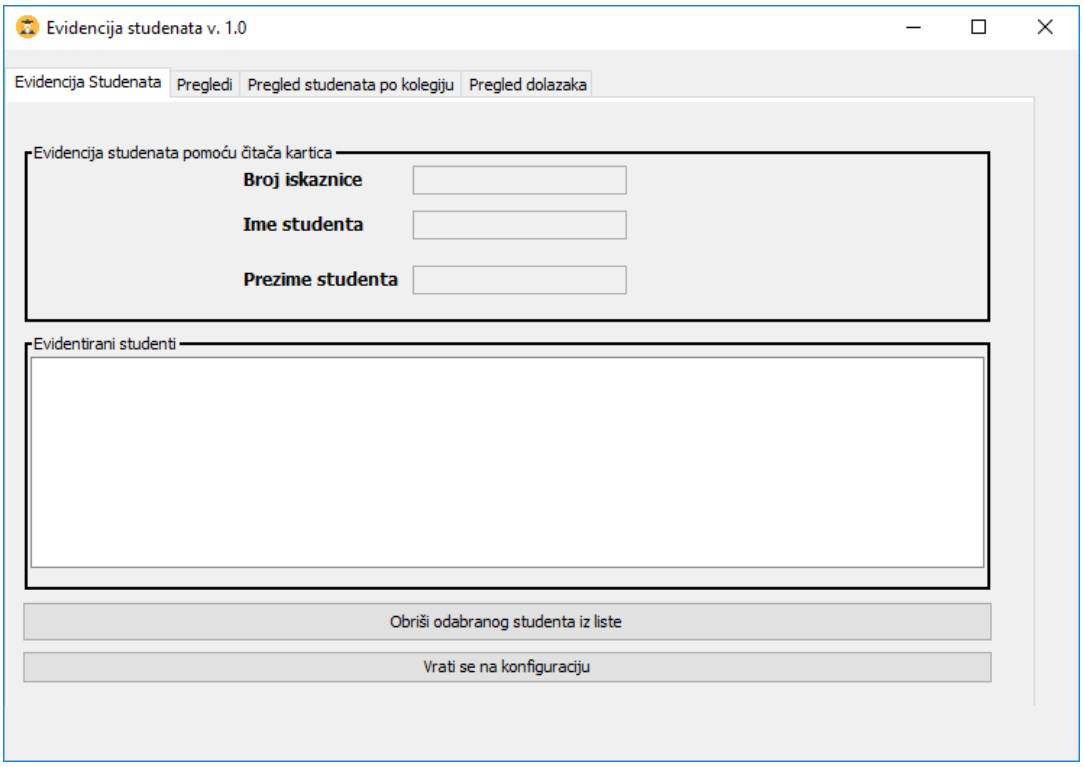
Slika 28. Prikaz profesorskog sučelja produkcijske desktop aplikacije 2



Izvor: Autor

Nakon pritiska na gumb „Potvrdi“ otvara se sučelje za evidentiranje studenata (Slika 29). Sučelje je sastavljeno od dvije sekcije. Prva sekcija sadrži osnovne podatke studenta (broj iskaznice, ime i prezime studenta) koji će se dobiti provlačenjem kartice studenta kroz čitač. Druga sekcija sadrži listu evidentiranih studenata. Lista sadrži studente koji su se evidentirali preko desktop aplikacije putem kartice (X-ice) ili web aplikacije gdje ih je evidentirao profesor u slučaju da su zaboravili svoju karticu. U sučelju postoje i dva gumba: „Obriši odabranog studenta iz liste“ i „Vrati se na konfiguraciju“. Prvi gumb služi da profesor može obrisat određenog studenta iz liste na način da ga odabere u listi i stine na gumb. Drugi gumb služi da se može vratiti na konfiguraciju ukoliko je nešto krivo odabrano npr. vrsta predavanja (Slika 28). Studenti mogu krenuti provlačiti svoje studentske kartice (X-ice) kroz čitač kad im profesor dozvoli.

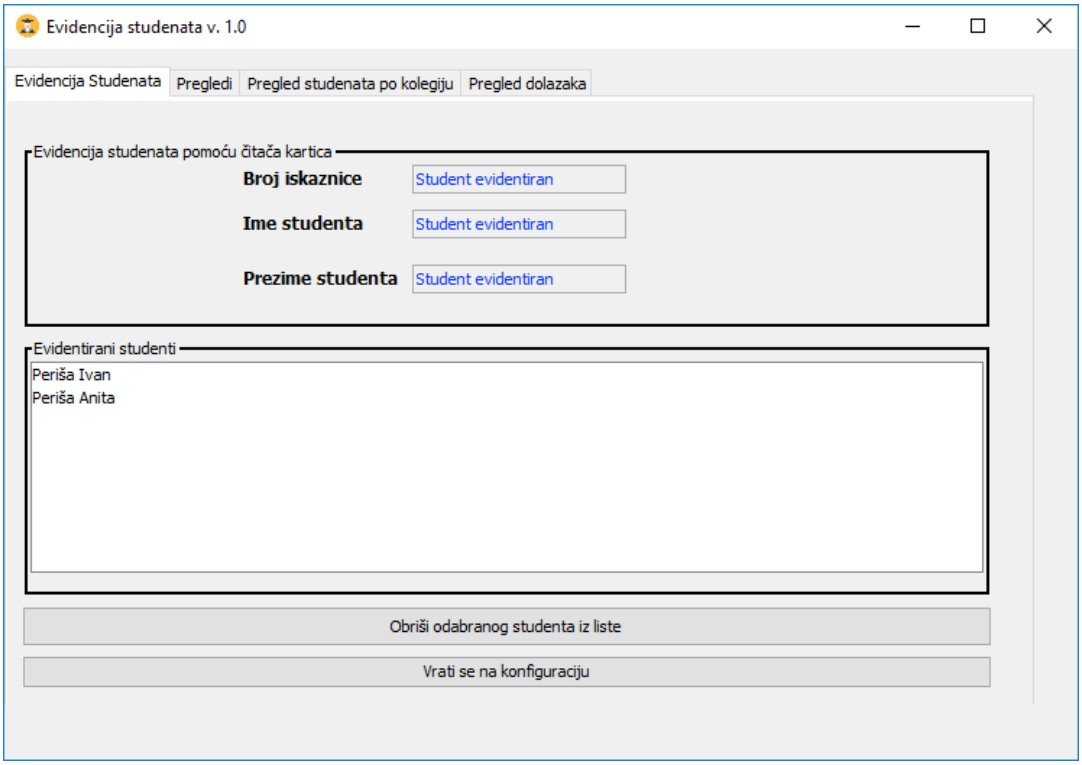
Slika 29. Prikaz sučelja za evidenciju studenata produkcijske desktop aplikacije 1



Izvor: Autor

Važno je napomenuti da se kartica može provlačiti u bilo kojem smjeru, ali se mora paziti da je okrenuta na pravilnu stranu. (stranu gdje je magnetska glava za čitanje). Drugim riječima, magnetska traka na studentskoj kartici mora biti okrenuta prema magnetskoj glavi na čitaču kako bi se uspješno pročitala informacija s trake, a to su podatci studenta. Student nakon što je provukao svoju karticu (X-icu) kroz čitač, automatski se šalju informacije iz čitača u sustav. Sustav provjerava podatke te javlja grešku ukoliko student nije upisan na kolegij za koji se evidentira ili ukoliko je već evidentiran za taj kolegij. U suprotnom sustav evidentira studenta u bazu podataka te ga automatski dodaje u listu u drugoj sekciji (Slika 30). Nakon evidencije svih studenata, profesor ima mogućnosti promjene prisustva istih. Ukoliko dođe do nekih nepravilnosti, profesor može obrisati tog studenta sa termina predavanja na gumb „Obriši odabranog studenta iz liste“. Na klik gumba otvara se skočni prozor koji zahtjeva da profesor još jedanput potvrdi da li je siguran da želi obrisati studenta s termina predavanja određenog kolegija.

Slika 30. Prikaz sučelja za evidenciju studenata produkcijske desktop aplikacije 2



Izvor: Autor

9. Zaključak

Sustav je izrađen u dvije aplikacije: desktop i web. U njihovom razvijanju važnu ulogu ima UML. UML model preko osnovnih dijagrama koji su se koristili kroz ovaj rad grafički prikazuje procese koji se trebaju izvršiti, što uvelike pomaže razvijatelju aplikacija u njihovom konstruiranju. Osim UML dijagrama važnu ulogu imaju klase koje formiraju temelje za objektno orijentirane aplikacije kao što su ove. Za izradu baze podataka važan je EVA model koji se kasnije transformira u relacijski model kako bi se kreirala baza podataka. Sve komponente kroz koje je sustav opisan su bitne za izgradnju aplikacija. Desktop aplikacija je izrađena u programskom jeziku Java, a web u programskom jeziku PHP (razvojni okvir Laravel). Aplikacije su napravljene pomoću MVC pristupa. Sustav povezuje sve načine evidentiranja studenata putem ove dvije aplikacije koje su opisane kroz rad. Glavna svrha mu evidentiranje studenata putem čitača kartica.

Literatura

1. Janssen, D. Relational model, Technopedia: https://www.techopedia.com/definition/24559/relational-model-database (5.6.2018.)
2. Miles, R., Hamilton, K., Learning UML 2.0, O'Reilly Media, Sebastopol, 2006.
3. Mileusnić, V., Izrada ERA modela baze podataka, Fakultet organizacije i informatike, Varaždin, 2011, https://moodle.srce.hr/eportfolio/artefact/file/download.php?file=31207&view=7460 (10.5.2018.)
4. Reed, P. R., Developing Applications with Java and Uml, Addison Wesley, Boston, 2001.
5. Seidl, M., UML @ Classroom. An Introduction to Object-Oriented Modeling, Springer, Heidelberg, 2012.
6. Taylor O., Laravel, https://laravel.com (20.5.2018.)
7. UML Use Case Diagrams, http://www.uml-diagrams.org/use-case-diagrams.html (1.4.2018.)

Popis slika

Slika 1. Prikaz arhitekture sustava 2
Slika 2. Prikaz dijagrama slučajeva korištenja web aplikacije 7
Slika 3. Prikaz dijagrama slučajeva korištenja desktop aplikacije 8
Slika 4. Prikaz dijagrama aktivnosti za unos, izmjenu i brisanje kolegija 11
Slika 5. Prikaz dijagrama aktivnosti za evidentiranje studenata preko čitača kartica 13
Slika 6. Prikaz početnog sučelja desktop aplikacije 14
Slika 7. Prikaz početnog administratorskog sučelja desktop aplikacije 15
Slika 8. Prikaz početnog profesorskog sučelja desktop aplikacije 16
Slika 9. Prikaz sučelja za evidentiranje studenata na desktop aplikaciji 17
Slika 10. Prikaz početnog sučelja web aplikacije 18
Slika 11. Prikaz početnoga administratorskog sučelja web aplikacije 19
Slika 12. Prikaz početnoga profesorskog sučelja web aplikacije 20
Slika 13. Prikaz web stranice za unos termina 21
Slika 14. Prikaz primjera klase 22
Slika 15. Prikaz dijagrama klase 23
Slika 16. Prikaz EVA modela 26
Slika 17. Prikaz relacijskog modela 29
Slika 18. Prikaz sekvencijalnog dijagrama za unos kolegija 30
Slika 19. Prikaz sekvencijalnog dijagrama za izmjenu kolegija 31
Slika 20. Prikaz sekvencijalnog dijagrama za brisanje kolegija 32
Slika 21. Prikaz sekvencijalnog dijagrama za evidentiranje studenata preko čitača kartica 33
Slika 22. Prikaz programskog koda migracije u Laravelu 35
Slika 23. Prikaz programskog koda klase modela "User" 36
Slika 24. Prikaz izgleda sučelja za unos korisnika na web aplikaciji 36
Slika 25. Prikaz programskog koda klase kontrolora " UsersController " 37
Slika 26. Prikaz početne stranice produkcijske desktop aplikacije 38
Slika 27. Prikaz profesorskog sučelja produkcijske desktop aplikacije 1 39
Slika 28. Prikaz profesorskog sučelja produkcijske desktop aplikacije 2 41
Slika 29. Prikaz sučelja za evidenciju studenata produkcijske desktop aplikacije 1 42
Slika 30. Prikaz sučelja za evidenciju studenata produkcijske desktop aplikacije 2 43