Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za računarstvo i informatiku

OPTIMIZACIJA RASPOREDA

Studentski projekat iz predmeta
Objektno orijentisana analiza i dizajn

Taida Kadrić
Nejra Sadžak
Faruk Smajlović

Sarajevo, 2020.

Sadržaj

1. Opis projekta 2
2. Dijagram slučajeva upotrebe i scenariji 3
2.1 Scenarij 1 4
2.2 Scenarij 2 5
2.3 Scenarij 3 6
2.4 Scenarij 4 7
2.5 Scenarij 5 8
2.6 Scenarij 6 9
2.7 Scenarij 7 10
2.8 Scenarij 8 11
3. Dijagrami aktivnosti 13
3.1 Dijagrami aktivnosti administratora 13
3.2 Dijagrami aktivnosti profesora 14
3.3 Dijagrami aktivnosti studenta 15
4. Klase 16
4.1 Dijagram i opis klasa 16
5. SOLID principi 18
6. Entity-Relationship dijagram 19
7. Dizajn paterni 20
7.1 Strukturalni paterni 20
7.2 Kreacijski paterni 21
7.3 Paterni ponašanja 23
8. MVC dijagram 26
9. Dijagram komponenti 28
10. Dijagrami sekvence 29
10.1 Dijagrami sekvence administratora 29
10.2 Dijagrami sekvence profesora 31
10.3 Dijagrami sekvence sudenta 32
11. Dijagrami složenih struktura 34
12. Forme aplikacije 35
13. Zašto izabrati ovu aplikaciju i kako je u budućnosti unaprijediti? 41

1. Opis projekta

Pri pokretanju aplikacije korisniku će se otvoriti login prozor, tj. bit će zatraženo unošenje korisničkog imena i lozinke. Ukoliko korisnik bude prepoznat kao student bit će mu ponuđeno generisanje rasporeda za predstojeći semestar, te potom izmjena i prihvatanje rasporeda. U slučaju da se loguje student koji je prethodno generisao i prihvatio raspored, neće mu biti ponuđeno ponovno generisanje, već će biti ispisan onaj raspored koji je student potvrdio.

Ukoliko se osoba koja koristi aplikaciju prijavi kao profesor, bit će u mogućnosti da odredi predmet, ukoliko predaje više predmeta, te termine predavanja i vježbi za svoj predmet vodeći računa o dostupnosti sala.
Što se tiče određivanja termina predavanja i vježbi, svaki profesor će trebati odrediti broj termina, vrijeme i salu održavanja kako bi svi student mogli prisustvovati (s obzirom na to da postoji ograničenje na broj studenata koji može biti na vježbama).

Nakon što su svi profesori odabrali termine tutorijala za svoje predmete student će imati mogućnost pokretanja optimizacije rasporeda na osnovu dodijeljenih predmeta. Osnovna ideja jedne takve optimizacije jeste da se samo jednim klikom dobije algoritamski generisan prijedlog rasporeda koji je za datog studenta vremenski najoptimalniji. To znači da će algoritam sastaviti raspored tako da student ima najmanje mogućih pauza između predavanja i drugih vježbi u toku sedmice. U slučaju da iz nekog razloga studentu ne odgovara taj generisani raspored biti će ostavljena opcija neke vrste ručnog stvaranja rasporeda. Nakon toga ostaje samo još prihvatanje datog rasporeda kao konačnog.

Studenti i profesori, ukoliko nisu zadovoljni dodijeljenim korisničkim imenom, lozinkom ili žele promijeniti polja sa imenom, prezimenom ili email-om, bit će u mogućnosti da naprave sve željene ispravke.

Administrator će biti u mogućnosti da dodaje nove studente, dodjeljuje im odsjek, semestar i predmete. Potom dodaje nove profesore i dodjeljuje im predmet ili više njih. U slučaju promjene stanja opreme u sali, omogućava mu se promjena kapaciteta sale ili dodavanje nove sale. U slučaju izmjene nastavnog plana i programa, administrator će moći izmijeniti, dodati ili izbrisati bilo koji od podataka baze.

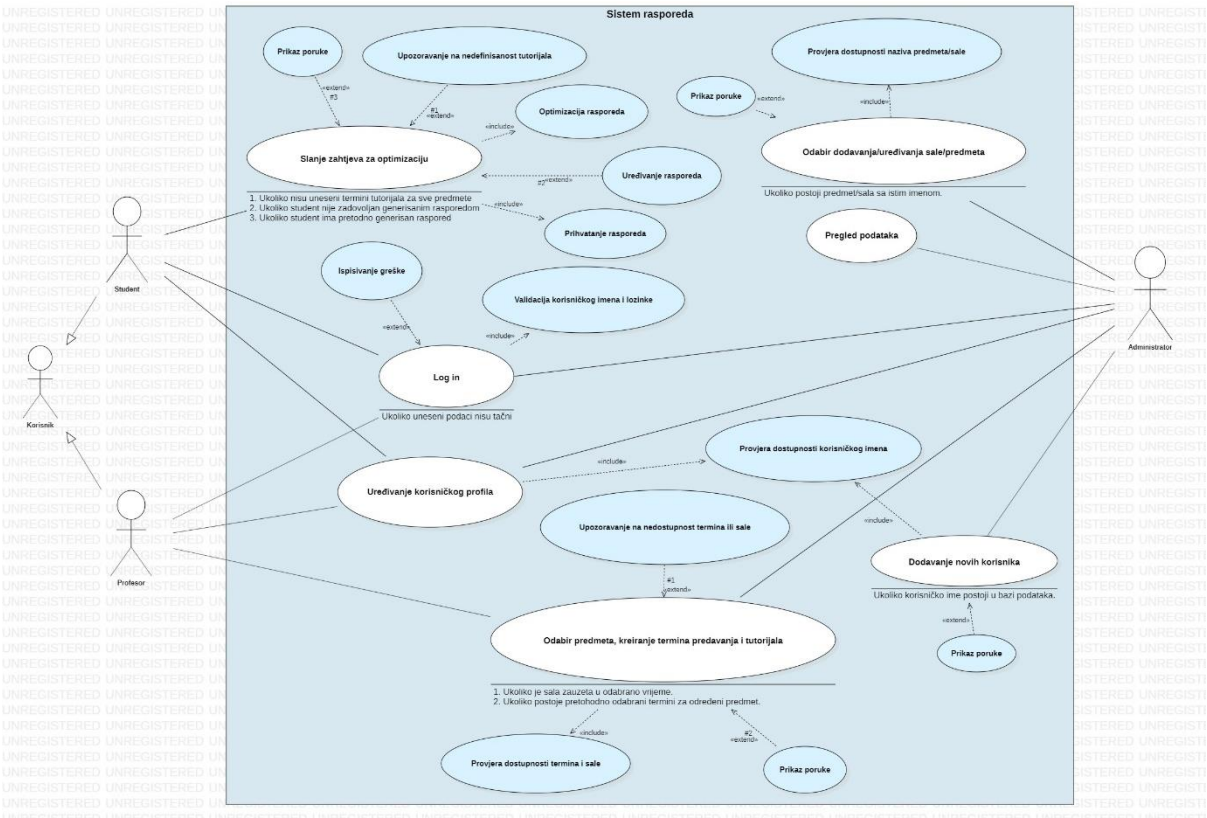
2. Dijagram slučajeva upotrebe i scenariji

Na slici ispod prikazan je dijagram slučajeva upotrebe (Use Case Diagram). Iz pomenutog dijagrama možemo vidjeti da u aplikaciji postoje tri vrste korisničkih profila, Student, Profesor i Administrator.

Administrator održava sistem, omogućava ostalim korisnicima pristup aplikaciji i ima uvid u sve podatke u bazi podataka.

Osnovna akcija profila profesora je kreiranje termina predavanja/vježbi za dodijeljeni predmet ili predmete. Navedena akcija uključuje sve potrebne provjere za ispravnost kreiranog termina kao što je dostupnost sale u odabrano vrijeme. Proširenje akcije kreiranja termina predstavljaju poruke koje se ispisuju u slučaju da su termini prethodno kreirani ili odabrana sala zauzeta.

Primarni cilj aplikacije je isporučivanje optimiziranog rasporeda stundetu, te je glavna akcija pridružena studentu slanje zahtjeva za optimizaciju rasporeda. Akcija slanja zahtjeva uključuje optimizaciju i prihvatanje rasporeda. Proširenje navedene akcije je pružanje mogućnosti za uređivanje ponuđenog rasporeda, te ispisivanje odgovarajućih poruka ukoliko raspored već postoji ili nisu definirani svi potrebni termini za optimiziranje.



Slika 1: Dijagram slučajeva upotrebe

Dijagram slučajeva upotrebe upotpunit ćemo uz detaljne scenarije svih aktivnosti prikazanih na prethodnom dijagramu.

2.1 Scenarij 1

Naziv: Prijava profesora na aplikaciju i odabir termina za tutorijale.

Opis: Profesor se prijavljuje na aplikaciju, bira predmet i dva termina tutorijala, u zavisnosti od dostupnosti sale u odabranom terminu.

Glavni tok: Uspješno određuje dva termina tutorijala za svoj predmet.

Preduvjeti: Profesor se uspješno prijavio na aplikaciju.

Posljedice: Profesor uspješno odabrao termine tutorijala.

Tok događaja:

Profesor	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir predmeta	
4. Odabir broja tutorijala (npr. 2)	
5. Odabir prvog termina i sale.	6. Provjera dostupnosti odabranog termina i sale.
7. Odabir drugog termina i sale.	8. Provjera dostupnosti odabranog termina i sale.
9. Spašavanje termina.	10. Spašavanje termina u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Profesor	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Zauzeta odabrana sala.

Preduvjeti: Na koracima 5 ili 7 odabrani termin ili sala su zauzeti.

Tok događaja:

Profesor	Sistem rasporeda
	1. Odabrani termin ili sala nisu dostupni.
	2. Upozoravanje korisnika da odabrani termin ili sala nisu dostupni.
	3. Vraćanje korisnika na ponovni odabir.
4. Nastavak od koraka 5/7 glavnog toka.	

Alternativni tok 3: Profesor je prethodno odabrao termine tutorijala.

Preduvjeti: U koraku 3 glavnog toka zaključeno je da je profesor prethodno odredio termine tuturijala.

Tok događaja:

Profesor	Sistem rasporeda
	1. Ispisivanje poruke.

2.2 Scenarij 2

Naziv: Prijava profesora na aplikaciju i odabir termina predavanja.

Opis: Profesor se prijavljuje na aplikaciju, bira predmet i tri termina predavanja, u zavisnosti od dostupnosti sale u odabranom terminu.

Glavni tok: Uspješno određuje tri termina predavanja za svoj predmet.

Preduvjeti: Profesor se uspješno prijavio na aplikaciju.

Posljedice: Profesor uspješno odabrao termine predavanja.

Tok događaja:

Profesor	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir predmeta	
4. Odabir broja predavanja (npr. 3)	
5. Odabir prvog termina i sale.	6. Provjera dostupnosti odabranog termina i sale.
7. Odabir drugog termina i sale.	8. Provjera dostupnosti odabranog termina i sale.
9. Odabir trećeg termina i sale.	10. Provjera dostupnosti odabranog termina i sale.
11. Spašavanje termina.	12. Spašavanje termina u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Profesor	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka..
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Zauzeta odabrana sala.

Preduvjeti: Na koracima 5, 7 ili 9 odabrani termin ili sala su zauzeti.

Tok događaja:

Profesor	Sistem rasporeda
	1. Odabrani termin ili sala nisu dostupni.
	2. Upozoravanje korisnika da odabrani termin ili sala nisu dostupni.
	3. Vraćanje korisnika na ponovni odabir.
4. Nastavak od koraka 5/7/9 glavnog toka.	

Alternativni tok 3: Profesor je prethodno odabrao termine predavanja.

Preduvjeti: U koraku 3 glavnog toka zaključeno je da je profesor prethodno odredio termine predavanja.

Tok događaja:

Profesor	Sistem rasporeda
	1. Ispisivanje poruke.

2.3 Scenarij 3

Naziv: Prijava profesora na aplikaciju i izmjena korisničkih podataka.

Opis: Profesor se prijavljuje na aplikaciju i želi izmijeniti svoje korisničke podatke (ime, prezime, korisničko ime, lozinku ili e-mail).

Glavni tok: Profesor uspješno izvršava izmjene na svom korisničkom profilu.

Preduvjeti: Profesor se uspješno prijavio na aplikaciju.

Posljedice: Profesor je uspješno izmijenio korisničke podatke.

Tok događaja:

Profesor	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir izmjene korisničkih podataka.	
4. Izmjena korisničkih podataka.	5. Provjera unesenih podataka.
6. Spašavanje izmjena.	7. Spašavanje izmjena u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Profesor	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Odabrano postojeće korisničko ime.

Preduvjeti: Na koraku 4 glavnog toka uneseno korisničko ime koje već postoji u bazi podataka.

Tok događaja:

Profesor	Sistem rasporeda
	1. Uneseni podaci nisu dostupni.
	2. Upozoravanje korisnika da željeno korisničko ime već postoji u bazi podataka.
	3. Vraćanje korisnika na ponovni unos podataka.
4. Nastavak od koraka 4 glavnog toka.	

2.4 Scenarij 4

Naziv: Prijava studenta na aplikaciju i generisanje optimiziranog rasporeda.

Opis: Student se prijavljuje na aplikaciju i šalje zahtjev za optimizaciju rasporeda.

Glavni tok: Student prihvata predloženi raspored.

Preduvjeti: Student se uspješno prijavio na aplikaciju.

Posljedice: Student dobija optimizovani raspored.

Tok događaja:

Student	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Slanje zahtjeva za optimizaciju rasporeda.	4. Provjera kompletnosti svih predmeta.
	5. Ispisivanje optimiziranog rasporeda.
6. Prihvatanje rasporeda i završavanje interakcije sa sistemom.	7. Spašavanje generisanog rasporeda.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Student	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Nemogućnost generisanja optimizvanog rasporeda zbog nekompletnosti predmeta (tj. profesor nije unio termine tutroijala).

Preduvjeti: U koraku 4, prilikom provjere dolazi informacija da nisu uneseni potrebni termini tutorijala.

Tok događaja:

Student	Sistem rasporeda
	1. Upozoravanje korisnika da nisu uneseni svi potrebni termini.
	2. Vraćanje studenta na ponovno slanje zahtjeva.

Alternativni tok 3: Student želi ručno promijeniti generisani raspored.

Preduvjeti: U koraku 5 student se odlučuje prema svojim obavezama promijeniti generisani raspored.

Tok događaja:

Student	Sistem rasporeda
1. Odabir alternativnih opcija tutorijala.	2. Prihvatanje izmijenjenog rasporeda.
3. Vraćanje korisnika na korak 6 glavnog toka.	

Alternativni tok 4: Student je prethodno generisao i prihvatio raspored.

Preduvjeti: U koraku 2 glavnog toka, uneseni podaci su validirani i zaključeno je da student ima pretohodno napravljen i potvrđen raspored.

Tok događaja:

Student	Sistem rasporeda
	1. Ispisivanje poruke.

2.5 Scenarij 5

Naziv: Prijava studenta na aplikaciju i izmjena korisničkih podataka.

Opis: Student se prijavljuje na aplikaciju i želi izmijeniti svoje korisničke podatke (ime, prezime, korisničko ime, lozinku ili e-mail).

Glavni tok: Student uspješno izvršava izmjene na svom korisničkom profilu.

Preduvjeti: Student se uspješno prijavio na aplikaciju.

Posljedice: Student je uspješno izmijenio korisničke podatke.

Tok događaja:

Student	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir izmjene korisničkih podataka.	
4. Izmjena korisničkih podataka.	5. Provjera unesenih podataka.
6. Spašavanje izmjena.	7. Spašavanje izmjena u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Student	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Odabrano postojeće korisničko ime.

Preduvjeti: Na koraku 4 glavnog toka uneseno korisničko ime koje već postoji u bazi podataka.

Tok događaja:

Student	Sistem rasporeda
	1. Uneseni podaci nisu dostupni.
	2. Upozoravanje korisnika da željeno korisničko ime već postoji u bazi podataka.
	3. Vraćanje korisnika na ponovni unos podataka.
4. Nastavak od koraka 4 glavnog toka.	

2.6 Scenarij 6

Naziv: Prijava administratora na aplikaciju.

Opis: Administrator se prijavljuje na aplikaciju, izvršava pregled i izmjene podataka.

Glavni tok: Administrator dodaje novi predmet/salu.

Preduvjeti: Administrator se uspješno prijavio na aplikaciju.

Posljedice: Dodan je novi predmet / nova sala.

Tok događaja:

Administrator	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir pregleda svih predmeta/sala.	
4. Dodavanje podataka o novom predmetu/sali.	5. Provjera dostupnosti naziva predmeta/sale.
6. Spašavanje unesenih podataka.	7. Spašavanje novih podataka u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Administrator	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Uneseni naziv predmeta/sale već postoji.

Preduvjeti: Na koraku 4 glavnog toka unesen je naziv predmeta/sale koji već postoji.

Tok događaja:

Administrator	Sistem rasporeda
	1. Uneseni naziv predmeta/sale nije dostupan.
	2. Upozoravanje korisnika na nedostupnost unesenih podataka.
	3. Vraćanje unos podataka.
4. Nastavak od koraka 4 glavnog toka.	

2.7 Scenarij 7

Naziv: Prijava administratora na aplikaciju.

Opis: Administrator se prijavljuje na aplikaciju, izvršava pregled i izmjene podataka.

Glavni tok: Administrator uređuje postojećeg studenta/profesora.

Preduvjeti: Administrator se uspješno prijavio na aplikaciju.

Posljedice: Izmijenjeni su podaci o postojećem studentu/profesoru.

Tok događaja:

Administrator	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir pregleda svih studenata/profesora.	
4. Uređivanje podataka o postojećem studentu/profesoru.	5. Provjera unesenih podataka.
6. Spašavanje unesenih podataka.	7. Spašavanje unesenih podataka u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Administrator	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.

	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.
4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Odabrano postojeće korisničko ime.

Preduvjeti: Na koraku 4 glavnog toka uneseno korisničko ime koje već postoji u bazi podataka.

Tok događaja:

Administrator	Sistem rasporeda
	1. Uneseni podaci nisu dostupni.
	2. Upozoravanje administratora da željeno korisničko ime već postoji u bazi podataka.
	3. Vraćanje na ponovni unos podataka.
4. Nastavak od koraka 4 glavnog toka.	

2.8 Scenarij 8

Naziv: Prijava administratora na aplikaciju.

Opis: Administrator se prijavljuje na aplikaciju, izvršava pregled i izmjene podataka.

Glavni tok: Administrator dodaje/uređuje termin predavanja/tutorijala.

Preduvjeti: Administrator se uspješno prijavio na aplikaciju.

Posljedice: Izmijenjeni su podaci o postojećem terminu predavanja/tutorijala.

Tok događaja:

Administrator	Sistem rasporeda
1. Pristupanje interfejsu za prijavu.	2. Validacija unesenih podataka.
3. Odabir pregleda svih termina predavanja/tutorijala.	
4. Dodavanje/uređivanje termina predavanja/tutorijala.	5. Provjera dostupnosti odabranog termina i sale.
6. Spašavanje unesenih podataka.	7. Spašavanje unesenih podataka u bazu podataka.

Alternativni tok 1: Neuspješna validacija korisničkih podataka.

Preduvjeti: Na koraku 1 glavnog toka uneseni su pogrešni podaci.

Tok događaja:

Administrator	Sistem rasporeda
	1. Korisnički podaci ne prolaze validaciju.
	2. Upozoravanje korisnika na neispravnost unesenih podataka.
	3. Vraćanje korisnika na prijavu.

4. Nastavak od koraka 1 glavnog toka.	

Alternativni tok 2: Odabrani termin/sala nije dostupan.

Preduvjeti: Na koraku 4 glavnog toka odabrani termin ili sala nisu dostupni.

Tok događaja:

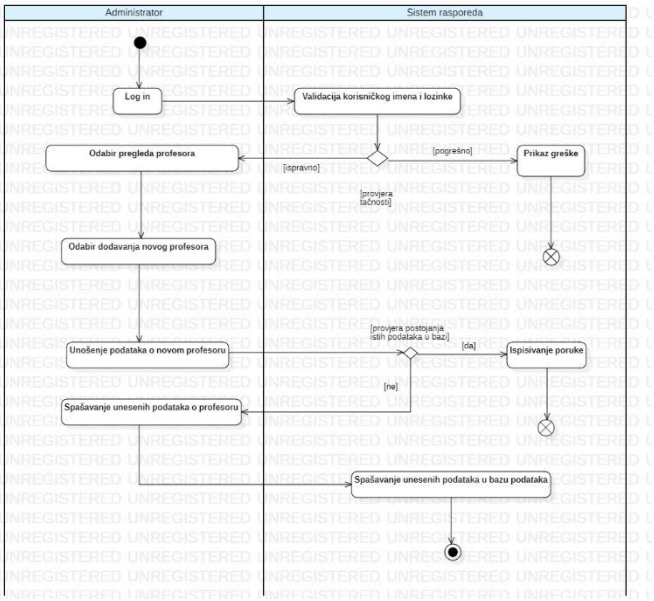
Administrator	Sistem rasporeda
	1. Uneseni podaci nisu dostupni.
	2. Upozoravanje administratora da su odabrani termin ili sala zauzeti.
	3. Vraćanje na ponovni odabir.
4. Nastavak od koraka 4 glavnog toka.	

3. Dijagrami aktivnosti

Dijagrami aktivnosti vežu se za prethodno navedene scenarije i omogućavaju nam procesni pogled na sistem. Podijeljeni su u tri skupine, u zavisnosti od profila osobe koja pristupa aplikaciji.

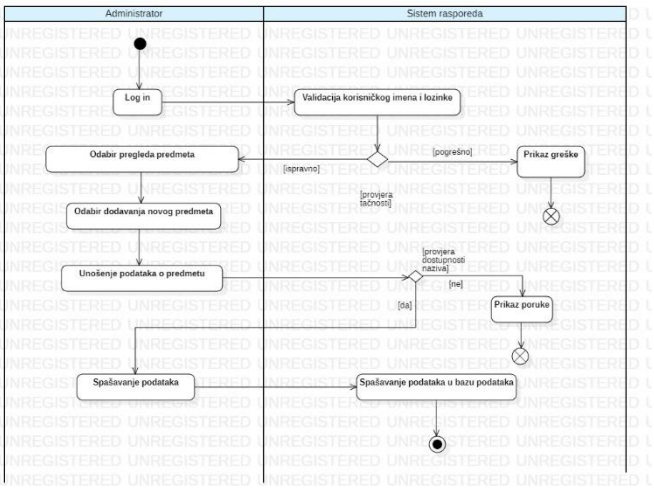
3.1 Dijagrami aktivnosti administratora

Dodavanje novih korisnika, sudenata ili profesora, od strane administratora vrši se na isti način. Razlika je u traženim podacima, npr. za studenta će se unositi odsjek i trenutni semestar, a za profesora titula. Na slici ispod prikazan je dijagram aktivnosti za dodavanje profesora.



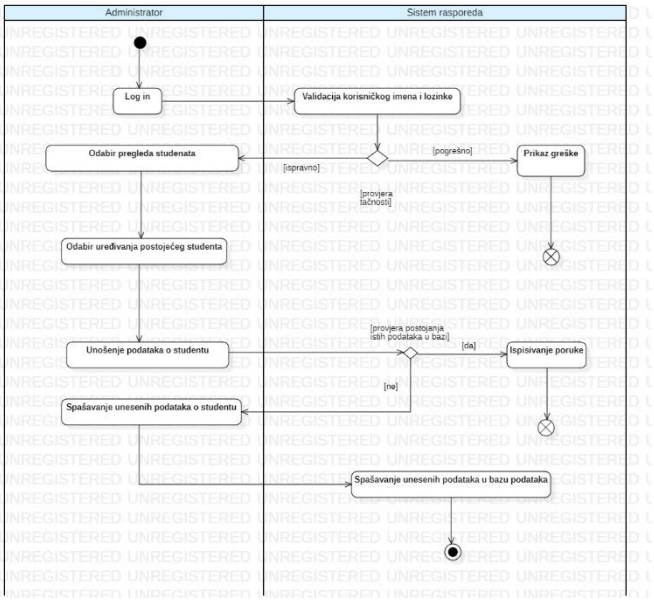
Slika 2: Dodavanje novog profesora

Na slici ispod prikazana je akcija dodavanja novog predmeta. Akcija dodavanja nove sale vrši se na identičan način, odnosno prisutne su provjere o dostupnosti naziva, obzirom da ne želimo dvije sale ili dva predmeta pod istim nazivom u bazi podataka.



Slika 3: Dodavanje novog predmeta

Uz navedena dodavanja korisnika, predmeta i sala, omogućene su i izmjene svih postojećih objekata u bazi podataka. Na slici ispod prikazan je dijagram aktivnosti promjene podataka o postojećem studentu.

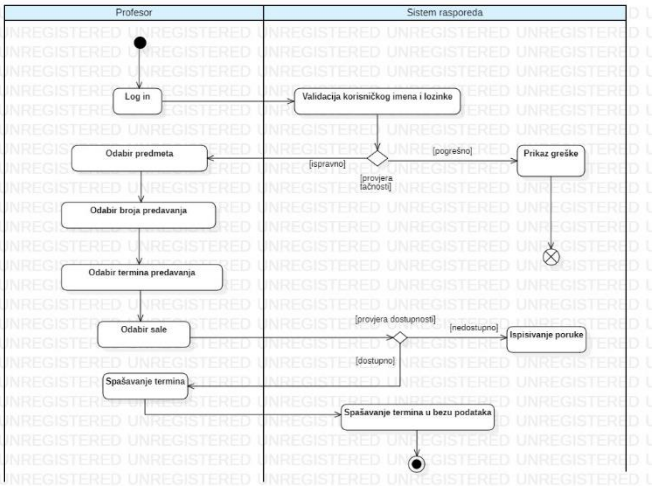


Slika 4: Izmjena postojećeg studenta

Akcija koja je također omogućena administratoru je dodavanje i uređivanje termina predavanja i vježbi koja se odvija na isti način kao profesorovog pristupa aplikaciji.

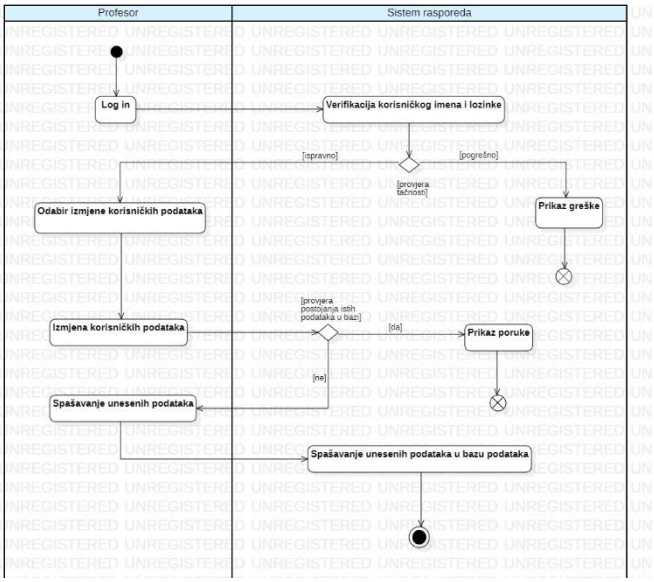
3.2 Dijagrami aktivnosti profesora

Kao što je prethodno navedeno, osnovna akcija profesora jeste kreiranje termina predavanja i tutorijala. Na slici ispod prikazan je dijagram aktivnosti kreiranja termina predavanja, na isti način vrši se kreiranje termina tutorijala.



Slika 5: Kreiranje termina predavanja/tutorijala

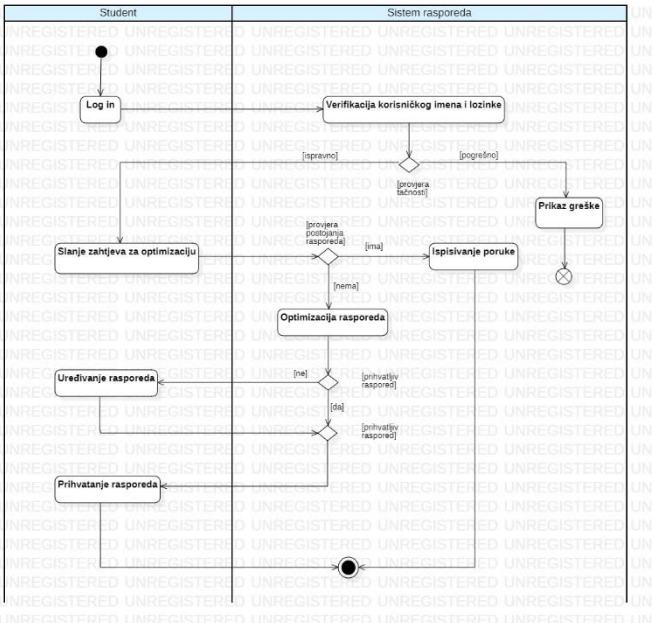
Sljedeća od aktivnosti ponuđenih profesoru je akcija uređivanja profila čiji se dijagram nalazi na slici ispod.



Slika 6: Uređivanje korisničkog profila

3.3 Dijagrami aktivnosti studenta

Aktivnosti ponuđene profilu studenta su slanje zahtjeva za optimizaciju i uređivanje svog korisničkog profila. Uređivanje profila vrši se na način prikazan u dijelu dijagrama aktivnosti profesora. Osnovna aktivnost studenta, slanje zahtjeva za optimiziranje rasporeda, prikazana je na dijagramu ispod.

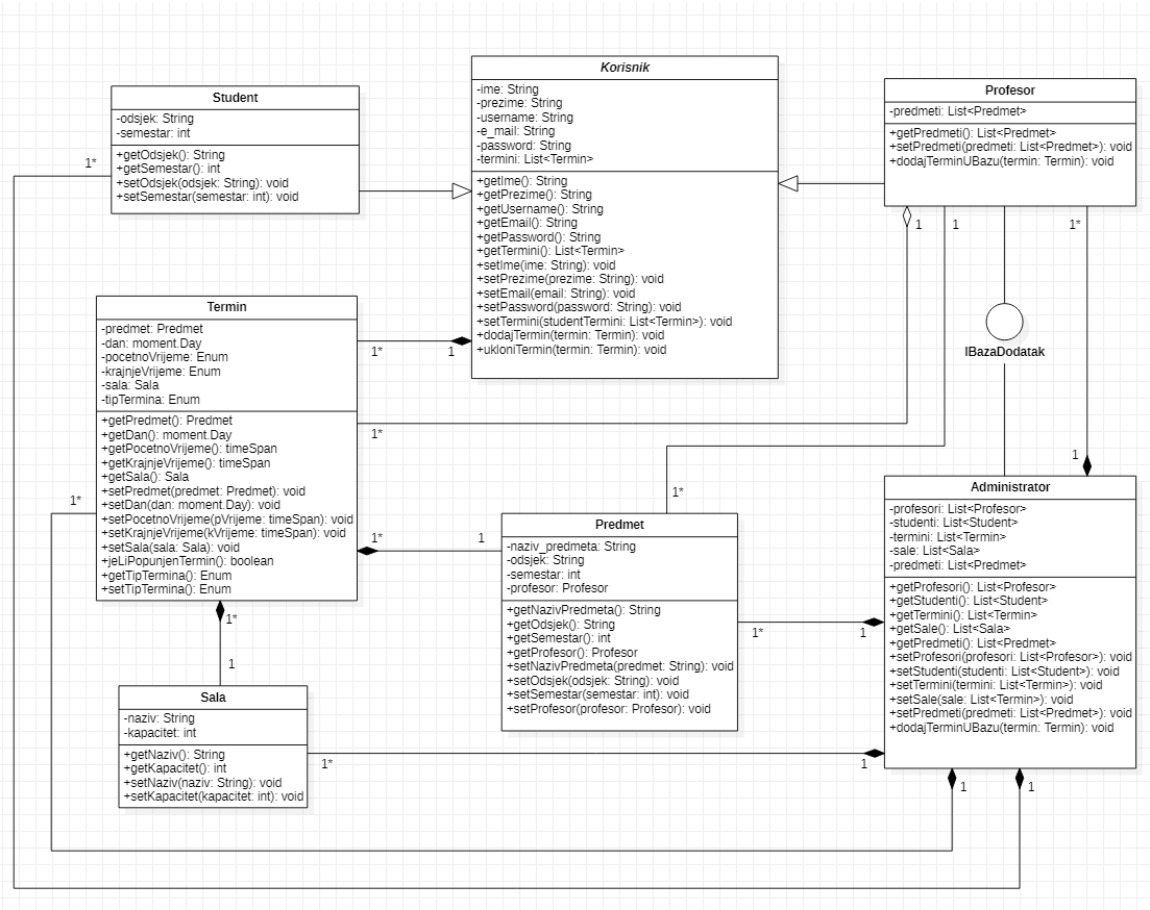


Slika 7: Slanje zahtjeva za optimizaciju

4. Klase

Iz do sada predstavljenog dijela uočene su potrebe za klasama: Korisnik, Student, Profesor, Termin, Predmet, Sala, Administrator.

4.1 Dijagram i opis klasa



Slika 8: Dijagram klasa

– Korisnik

Klasa Korisnik ima attribute: ime (String), prezime (String), username (String), eMail (String), password (String) i termini (List<Termin>). Metode ove klase su, pored setter i gettera, dodajTermin (void funkcija koja kao parameter prima objekat tipa Termin) i ukloniTermin (void funkcija koja kao parameter prima objekat tipa Termin).

– Student

Klasa Student izvedena je iz klase Korisnik. Dodatni atributi koje ova klasa ima su odsjek (String) i semestar (int). Atribut termini, klase Korisnik, bit će popunjeni terminima predavanja

i tutorijala predmeta koji su dodijeljeni određenom studentu. Metode ove klase su, pored metoda naslijeđenih iz klase Korisnik, potrebni getteri i setter.

– Profesor

Klasa Profesor izvedena je iz klase Korisnik. Dodatni atribut ove klase je predmeti (List<Predmet>), obzirom da jedan professor može predavati više predmeta. Atribut termini, klase Korisnik, bit će popunjeni terminima predavanja i tutorijala koje profesor bude kreirao za svoje predmete. Pored settera i gettera za listu predmeta, klasa Profesor ima metodu dodajTerminUBazu (void funkcija čiji je parametar objekat tipa Termin).

– Predmet

Atributi klase Predmet su: nazivPredmeta (String), odsjek (String), semestar (int) i profesor (Profesor). Metode ove klase su svi potrebni setteri i getteri.

– Termin

Atributi klase Termin su: predmet (Predmet), dan (moment.Day), pocetnoVrijeme (Enum), krajnjeVrijeme (Enum), sala (Sala), tipTermina (Enum). Pored gettera i settera, metoda klase Termin je jeLiPopunjenTermin (Boolean funkcija bez parametara).

Atribut tipTermina je atribut prema kojem će se razlikovati termini predavanja i tutorijala, obzirom da nema potrebe za izvođenjem klasa Predavanje i Tutorijal iz klase Termini.

– Sala

Klasa Sala sadrži atribute naziv (String) i kapacitet (int), a od metoda settere i gettere za navedene atribute. Korištena je u klasi Termin i pomoću metode jeLiPopunjenTermin provjeravat će se da li je specifični termin predavanja ili tutorijala popunjen, tj. da li kapacitet sale u kojoj se termin podržava nove studente.

– Administrator

Atributi klase Administrator su: profesori (List<Profesor>), studenti (List<Student>), termini (List<Termin>), sale (List<Sala>), predmeti (List<Predmet>). Pored gettera i settera za navedene atribute, klasa Administrator ima metodu dodajTerminUBazu (void funkcija čiji je parametar objekat tipa Termin).

Sve prethodno navedene klase korištene su u klasi Administrator obzirom da administrator ima uvid u sve podatke u bazi podataka i mogućnosti dodavanja novih, kao i uređivanje postojećih, profesora, studenata, termina, sala i predmeta.

Potpun i izmijenjen dijagram klasa biti će prikazan nakon razmotrenih i dodanih potrebnih dizajn paterna u dijelu MVC dijagrama.

5. SOLID principi

- Single Responsibility Principle

Odgovornost navedenog principa je da svaka od klasa korištenih u određenoj aplikaciji ima isključivo jednu svrhu. Ukoliko neka od klasa izvršava previše različitih tipova akcija u aplikaciji, ovaj princip to tumači kao „razlog za promjenu“.

Iz prethodno navedenog opisa klasa zaključujemo da je pomenuti princip zadovoljen.

- Open/Closed Principle

OC princip savjetuje dizajn sistema takav da buduće promjene ne uzrokuju mnogo modifikacija. Moguće buduće promjene su npr. dodavanje klase Asistent izvedene iz klase Korisnik, uvođenje dodatnih specifikacija koje će zahtijevati razlikovanje termina predavanja i tutorijala, samim tim izvođenje klasa Predavanje i Tutorijal iz klase Termin ili uvođenje izbornih predmeta, te izvođenje takve klase iz trenutne klase Predmet.

Sve navedene promjene neće znatno promijeniti dosadašnji dizajn Sistema te zaključujemo da je navedeni princip zadovoljen. Sve eventualne promjene desit će se u klasi Administrator koja jedino ima pristup svim podacima koji se nalaze u bazi.

- Liskov Substitution Principle

Liskov Substitution princip zahtijeva da nasljeđivanje bude ispravno implementirano, odnosno da je na svim mjestima na kojima se koristi osnovni objekat moguće iskoristiti i izvedeni objekat, a da takvo nešto ima smisla.

Navedeni pricip jeste zadovoljen prema trenutim definicijama klasa obzirom da je jedino nasljeđivanje nasljeđivanje iz klase Korisnik koja je apstraktna. Međutim, obje izvedene klase, Student i Profesor, imaju upotpunosti različite pristupe i akcije u sistemu te nikada neće doći do upotrebe bilo koje od ove dvije klase na mjestima gdje je korišten osnovni objekat.

- Interface Segregation Principle

Princip Interface Segregation zahtijeva da i svi interfejsi zadovoljavaju princip Single Responsibility, odnosno da svaki interfejs obavlja samo jednu vrstu akcija.

Interfejs korišten u sistemu je IBazaDodatak koji koriste klase Profesor i Administrator, a jedinja njegova akcija je ispravno dodavanje termina u bazu podataka. Zaključujemo da je navedeni princip zadovoljen.

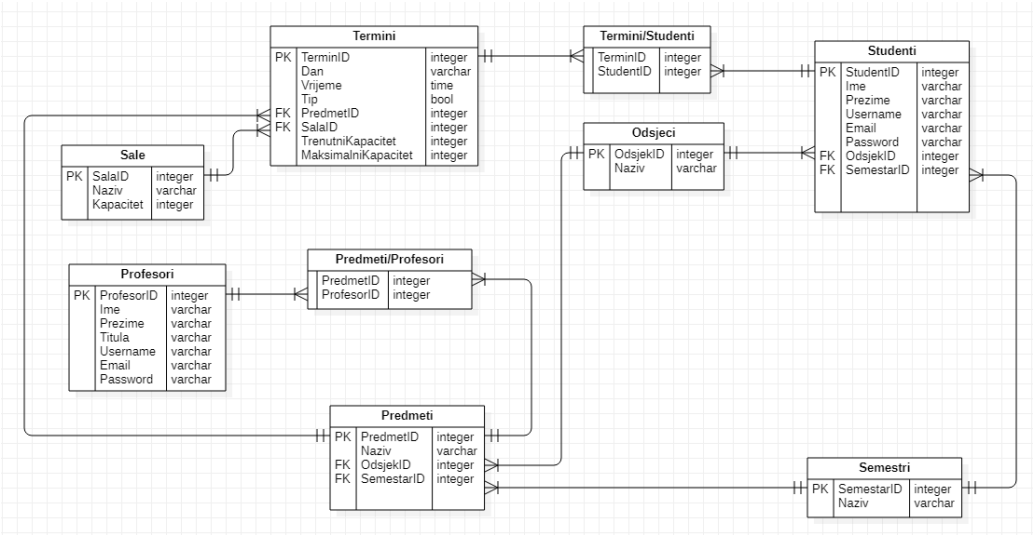
- Dependency Inversion Principle

Dependency Inversion princip zahtijeva da pri nasljeđivanju od strane više klasa bazna klasa uvijek bude apstraktna.

Jedino nasljeđivanje prisutno u sistemu je nasljeđivanje klasa Student i Profesor iz klase Korisnik koja je apstraktna, pa zaključujemo da je spomenuti princip zadovoljen.

6. Entity-Relationship dijagram

Na osnovu opisanih funkcionalnosti koje će aplikacija imati, uočene su potrebe za bazom podataka koja će imati sljedeće tabele: Termini, Studenti, Termini/Studenti, Sale, Profesori, Predmeti, Predmeti/Profesori, Odsjeci i Semestri.



Slika 9: ER dijagram

Detaljnije ćemo opisati neke od navedenih tabela.

Tabela Termini u sebi sadrži atribute TrenutniKapacitet i MaksimalniKapacitet kako bismo napravili manje opterećenje na bazu podataka pri dobavljanju raspoloživih termina za kreiranje optimiziranog rasporeda.

Tabela Termini/Studenti se popunjava za određenog studenta nakon što on prihvati svoj raspored kako bismo u svakom trenutku mogli vidjeti koji je student na kojem terminu.

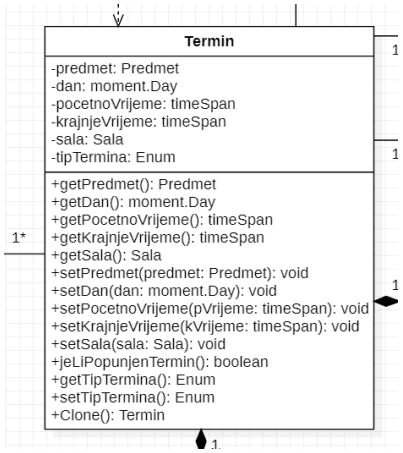
Tabela Predmeti/Profesori je kreirana u svrhu podržavanja više profesora na određenom predmetu ili više predmeta za određenog profesora.

Ostale tabele su prilično intuitivne (npr. Sudent ili Profesor), a ostale (npr. Sale, Semestri ili Odsjeci) su formirane zbog smanjivanja opterećenja na pretohodno navedene tabele, nad kojima će se češće vršiti upiti, a čiji podaci nisu od koristi u navedenim upitima.

7. Dizajn paterni

7.1 Strukturalni paterni

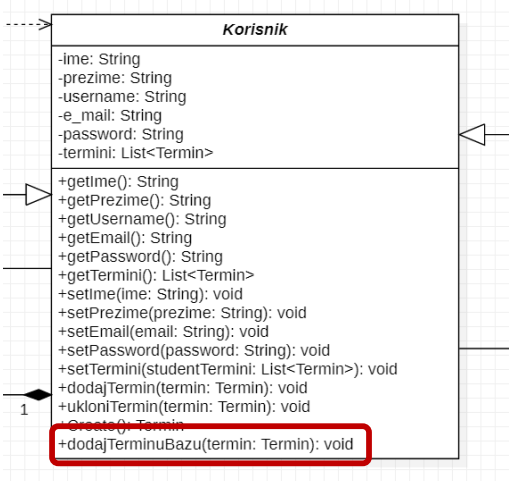
1. Adapter patern služi da se postojeći objekat prilagodi za korištenje na neki novi način u odnosu na postojeći rad, bez mijenjanja same definicije objekta. Obzirom na svrhu naše aplikacije, svi mogući budući dodaci ni na koji način neće moći koristiti trenutne funkcionalnosti bez narušavanja definicija klasa.
   Zaključujemo da navedeni patern nije primjenjih u našem projektu.
2. Fasadni patern služi kako bi se klijentima pojednostavilo korištenje kompleksnih sistema. Klijenti vide samo fasadu, odnosno krajnji izgled objekta, dok je njegova unutrašnja struktura skrivena.
   Klasa Termin će interno sadržavati veliku količinu provjera, obzirom da nije dozvoljeno selektovanje sale koja je u odabrano vrijeme pretodno zauzeta ili npr. ukupan broj studenata na određenom predmetu mora biti manji ili jednak od kapaciteta sale koja je odabrana za predavanja s obzirom na to da svi studenti moraju prisustvovati predavanjima, dok je obrnuta logika za termine tutorijala jer svaki student prisustvuje samo jednom takvom terminu. Spomenuti patern osigurava da o svim ovim provjerama profesor ili administrator, pri kreiranju termina, ne moraju voditi računa.



Slika 10. Fasadni patern

3. Decorator patern služi za omogućavanja različitih nadogradnji objektima koji svi u osnovi predstavljaju jednu vrstu objekta (odnosno, koji imaju istu osnovu).
   Navedeni patern bismo hipotetički mogli primijeniti na klasu termin, ukoliko bismo iz nje izveli dvije vrste termina, termini za predavanja i termini za tutorijale. Međutim, u našem sistemu ne postoji razlika između te dvije vrste termina te navedeni patern neće biti iskorišten.
4. Bridge patern služi kako bi se apstrakcija nekog objekta odvojila od njegove implementacije. Navedeni patern bi također mogao biti iskorišten u slučaju izvedenih klasa iz klase Termin te da se implementacije termina predavanja i tutorijala razliku.

5. Comoposite patern spada u skupinu strukturalnih paterna i koristi se za kreiranje hijerarhije objekata, tačnije za objekte koji imaju različite implementacije metoda kojima se pristupa na isti način. U našem projektu nema vidljive primjene ovog paterna.
6. Proxy patern je patern koji osigurava objekte od pogrešne ili zlonamjerne upotrebe i omogućava kontrolu pristupa. Ukoliko bismo našu aplikaciju nadogradili tako da i asistenti imaju mogućnost dodavanja termina tutorijala, a onemogućili im dodavanje termina predavanja, tada bismo shodno tome da li je osoba profesor ili asistent dali odgovarajuće pravo pristupa na dodavanje termina.
   U našoj aplikaciji Proxy patern je iskorišten na način da je klasi Student, izvedenoj iz klase Korisnik, onemogućeno dodavanje tutorijala u bazu podataka, dok je ta metoda dozvoljena klasi Profesor, također izvedenoj iz klase Korisnik.
   Pomenuti patern je iskorišten i pri optimizaciji studentskog rasporeda, tj. pri optimizaciji i uređivanju rasporeda bit će uzeti u obzir samo predmeti dodijeljeni studentu.

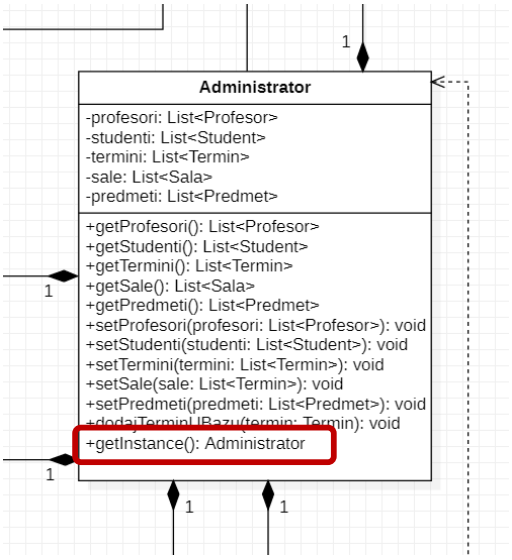


Slika 11: Proxy patern

7. *Flyweight* patern koristi se kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji svi u suštini predstavljaju jedan objekat. Samo ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama (tzv. specifično stanje), vrši se njegova instantacija, dok se u svim ostalim slučajevima koristi postojeća opća instanca objekta (tzv. bezlično stanje).
   Pomenuti patern ćemo primijeniti u našoj aplikaciji na način da ne pravimo duboke kopije sala koje se pridružuju terminima pri kreiranju termina predavanja/vježbi.

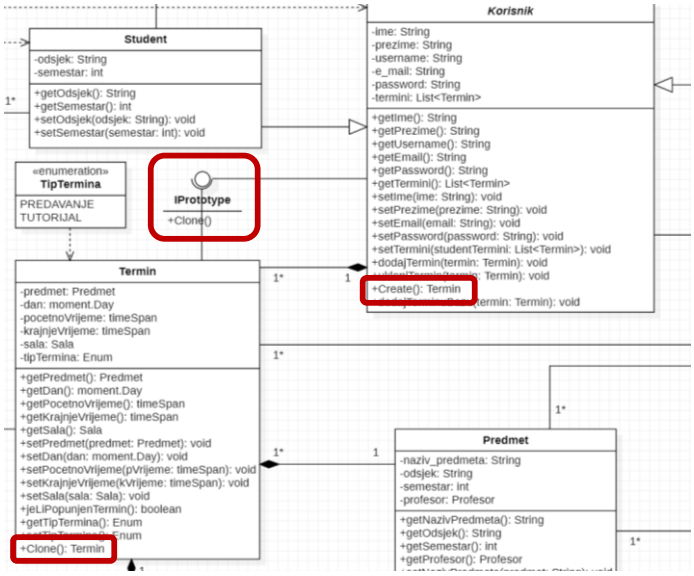
7.2 Kreacijski paterni

1. Singleton patern je jedan od kreacijskih paterna i koristi se kada se neka klasa instancira samo jednom.
   Što se tiče naše aplikacije, klasa administrator može biti singleton klasa s obzirom na to da je administrator jedna osoba koja se veoma rijetko mijenja, a ima interakciju sa svim podacima u bazi.



Slika 22. Singleton patern

2. Prototype patern omogućava smanjenje kompleksnosti kreiranja novog objekta tako što se uvodi operacija kloniranja. Na taj način prave se prototipi objekata koje je moguće replicirati više puta a zatim naknadno promijeniti jednu ili više karakteristika, bez potrebe za kreiranjem novog objekta nanovo od početka. Prototype patern bit će iskorišten pri kreiranju više termina predavanja/vježbi od strane profesora, obzirom na to da će se takvi termini razlikovati samo u vremenu održavanja ili sali.



Slika 13: Prototype patern

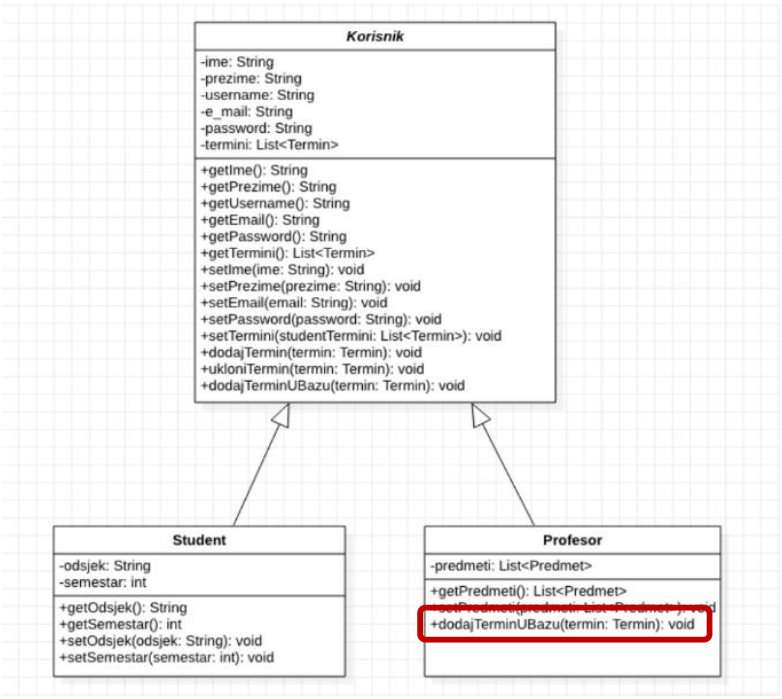
3. Factory method patern služi za omogućavanje instanciranje različitih vrsta podklasa koristeći factory metodu koja odlučuje koja će se podklasa instancirati i koja

programska logika izvršiti. Pogodnu primjenu Factory method paterna u našem projektu nismo pronašli.

4. Abstract factory patern služi kako bi se izbjeglo korištenje velikog broja if-else uslova pri kreiranju različitih hijerarhija objekata. Ukoliko postoji više tipova istih objekata te različite klase koriste različite podtipove, te klase postaju fabrike za kreiranje objekata zadanog podtipa bez potrebe za specificiranjem pojedinačnih objekata.
   Pomenuti patern nije iskorišten u našoj aplikaciji.
5. Builder patern služi za apstrakciju procesa konstrukcije objekta, kako bi se kao rezultat mogle dobiti različite specifikacije objekta koristeći isti proces konstrukcije. Ovaj patern koristi se kako bi se izbjeglo kreiranje kompleksne hijerarhije klasa te kako bi se izbjegao kompleksni programski kod konstruktora jedne klase koja može imati različite konfiguracije atributa.
   Builder patern neće biti iskorišten u našem projektu. Hipotetički, mogli bismo ga iskoristiti u slučaju da napravimo klase izvedene iz klase Termin, tj. klase Predavanje i Tutorijal.

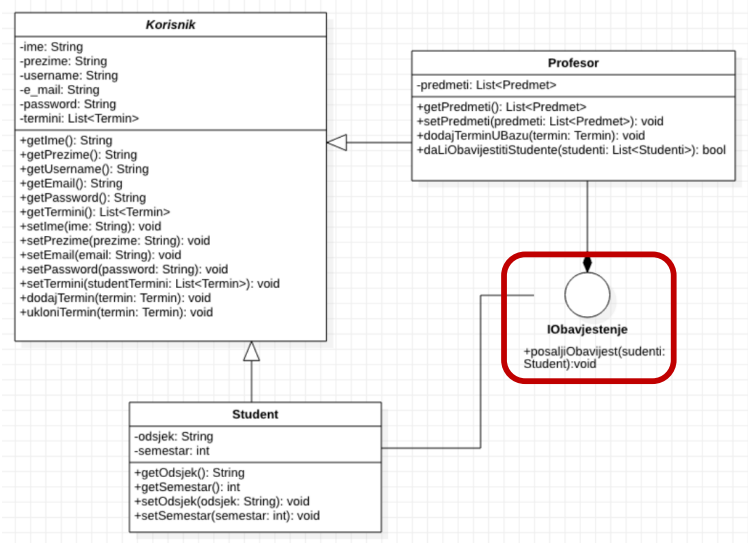
7.3 Paterni ponašanja

1. Strategy patern služi za razdvajanje različitih implementacija istog algoritma u različite klase.
   Ukoliko bismo imali metodu optimiziraj koja bi optimizirala raspored upotrebom različitih algoritama, tada bismo uveli interfejs IOptimizacija koji bi imao metodu optimiziraj. Tu metodu bi implementirale klase algoritama za optimizaciju (npr. ako bismo primijenili Ford Fulkersonov, Dijkstrin i Bellman Fordov algoritam). U slučaju da aplikacija utvrdi da je jedan algoritam efikasniji u odnosu na druge, s obzirom na broj studenata po semestru, ona će iskorititi određeni algoritam.
2. State patern se koristi kada se želi omogućiti objektu da mijenja svoja stanja od kojih zavisi njegovo ponašanje. U zavisnosti od stanja, čini se kao da je objekat promijenio klasu, a ta stanja se mijenjaju automatski. U našem projektu se ne mogu naći primjene ovog paterna.
3. Template Method patern služi za izmjenu ponašanja u jednom ili više dijelova. Template method patern je već primijenjen i to u dijelu kada imamo apstraktnu klasu Korisnik iz koje su naslijeđene Student i Profesor.
   U klasi Korisnik imamo metodu dodajTerminUBazu koju u suštini samo profesor koristi tj. sudent nema opciju dodavanja i uklanjanja termina u bazu, a profesor ima.



Slika 14: Template Method patern

4. Observer patern služi za kreiranje mehanizma pretplaćivanja. Pretplatnici dobiju obavještenje o sadržajima za čije slanje je zaslužna nadležna klasa. Ako bismo u našem projektu dodali funkcionalnost da nakon što su svi profesori unijeli termine predavanja i tutorijala studentima dođe obavijest (npr. na mail) o tome da mogu dobiti optimizirani raspored, tada bismo iskoristili ovaj patern. U klasi Profesor bi postojala metoda daLiObavijestitiStudente u kojoj bi se vršila provjera je li taj profesor posljednji u tom semestru koji je dodao potrebne termine predavanja i tutorijala i na osnovu toga bi se slalo obavještenje svim studentima u datom semestru da je njihova optimizacija spremna.



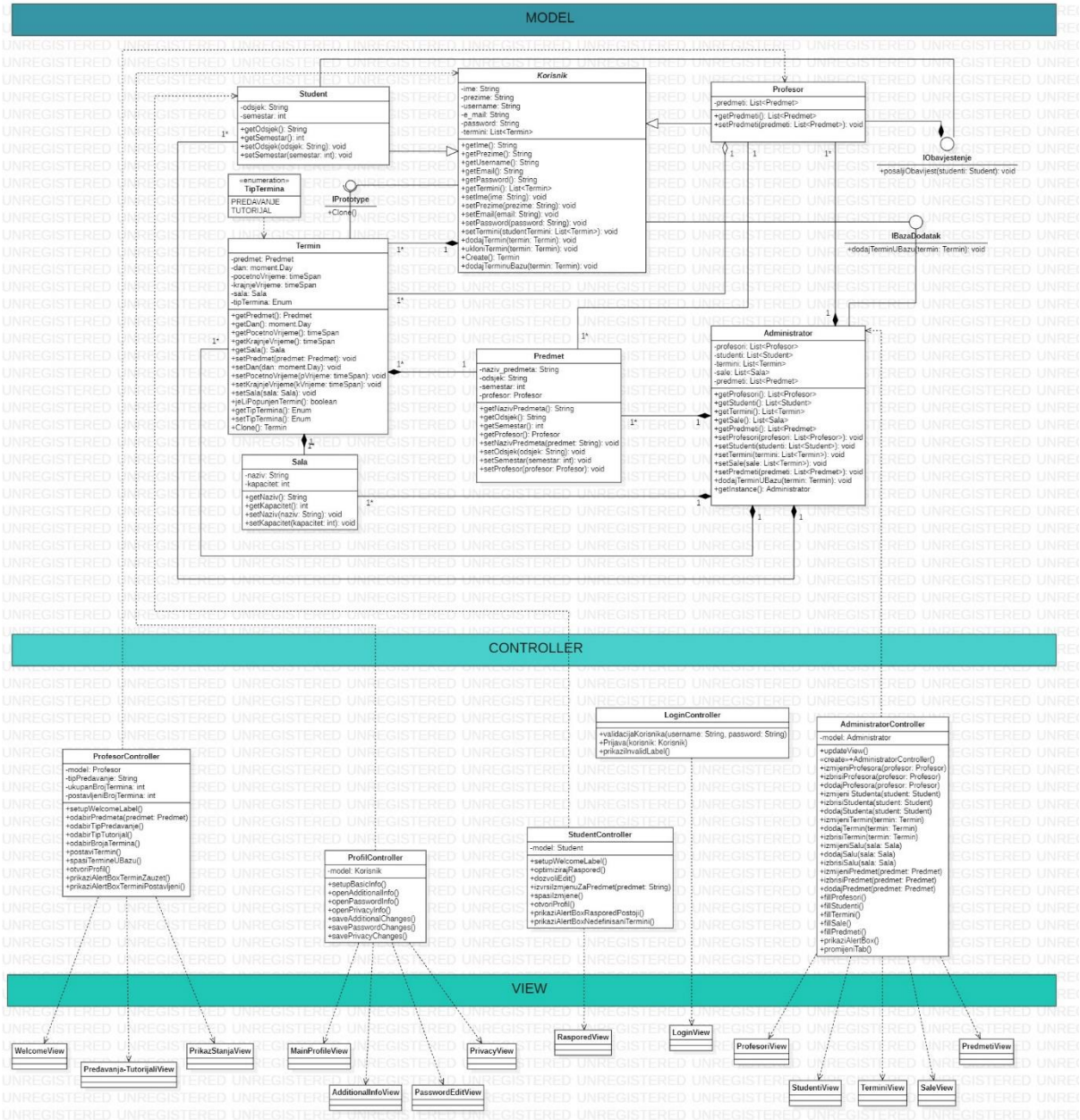
Slika 15: Observer patern

5. Iterator patern služi za omogućavanje prolaska kroz listu elemenata bez da su poznati implementacijski detalji struktrue u kojoj se čuvaju elementi. U našem projektu nemamo vidljivu primjenu ovog paterna.
6. Chain of Responsibility patern se koristi da se kompleksi procesi obrade razdvoje tako da više objekata na različite načine procesira primljene podatke. Realizuje se tako što svaki objekat koji vrši parcijalnu obradu prosljeđuje idućem objektu u nizu čime se smanjuje kompleksnost individualnih procesa i svaki objekat dobije samo podatke koji su mu potrebni. S obzirom na to da u našoj aplikaciji nema potrebe za automatskim dodavanjem određenih dijelova, ovaj patern nije primjenjiv na našu aplikaciju.
7. Mediator patern se koristi u svrhu manjenja broja veza između objekata. Objekti se ne spajaju direktno već uz pomoć medijatora koji je zadužen za komunikaciju. Pomoću njega kada jedan objekat želi proslijediti podatke drugom, to se realizuje preko medijatora koji odlučuje da li će proslijediti te podatke s obziorm na zadovoljenost određenih uslova.

   Ukoliko bismo u našoj aplikaciji uveli da i asistent može dodavati termine, ali samo tutorijala, a ne i predavanja onda bismo imali interfejs IMedijator koji bi imao metode provjeri_ koja provjerava da li je osoba student – ne može dodavati nikakve termine, asistent – može dodavati termine samo tutorijala ili profesor – može dodavati termine tutorijala i predavanja.

8. MVC dijagram

Na slici 16 prikazan je Mode-Controller-View dijagram.



Slika 36: MVC dijagram

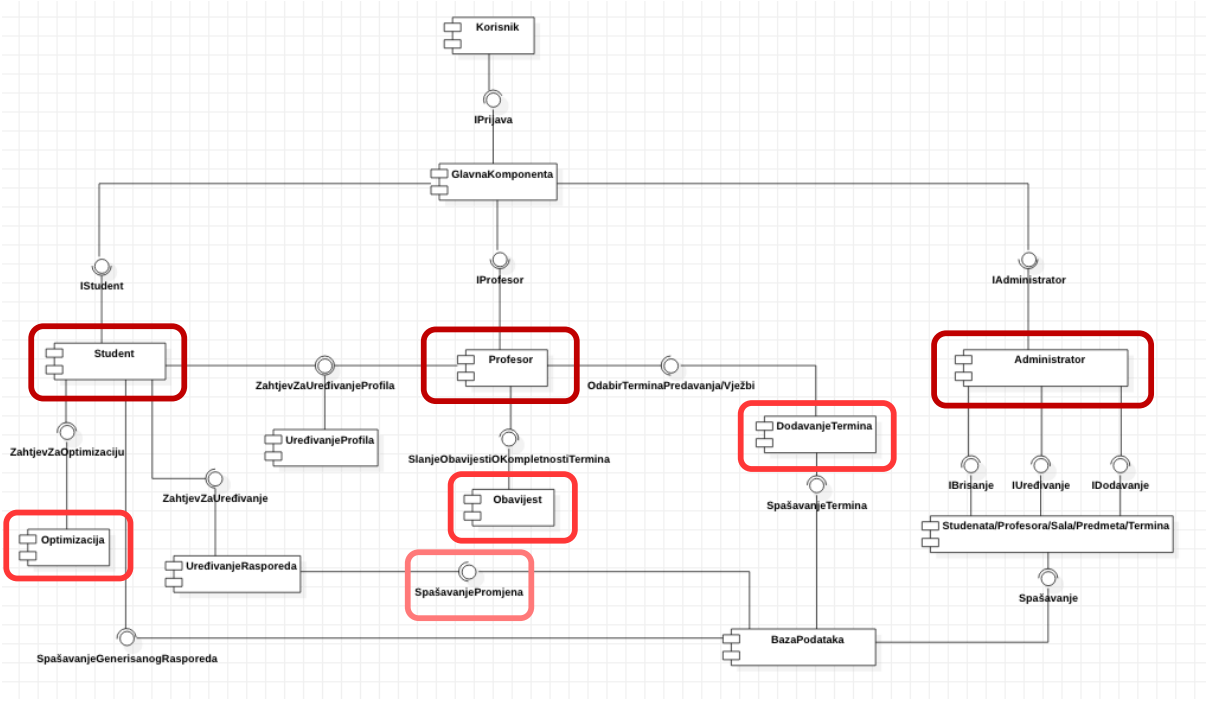
U dijelu modela prikazan je finalni dijagram klasa nakon razmotrenih dizajn paterna.

Dalje ćemo se osvrnuti na kontrolere. Metode koje imaju iste ili slične svrhe u različitim kontrolerima su: SetupWelcomeLabel koja na osnovu podataka iz baze ispiše Greeting poruku na ekranu nakon pravilnog logiranja u sistem. Još jedna takva metoda je OtvoriProfil() koja otvori profil trenutno logiranog korisnika sa njegovim osnovnim podacima. Obje ove metode se nalaze unutar ProfesorControllera i StudentControllera.

Također, obzirom da postoji više različitih AlertBox-ova nazvani su po imenima greške koje ispisuju. Jedina njihova razlika je String koji se ispiše na ekranu za izazvanu grešku.
ProfesorController ima metode koje su potrebne za odabir predmeta, predavanja i tutorijala kao i broja istih nakon čega ima opciju spašavanja svih termina u bazu podataka.
View-ovi koji se vežu za ProfessorController su početni view, view u kojem dodaje tutorijale i predavanja i posljednji view gdje vidi rezultate svih tih dodavanja na sedmičnom rasporedu.
ProfilController ima metode koje služe za editovanje standardnih informacija u korisniku kao i otvaranje view-ova koji prikazuju te informacije a to su AdditionalInfo, Privacy i Password view.
StudentController sadrži metode OptimizirajRaspored koja služi za pokretanje algoritma optimizacije rasporeda, nakon što se raspored interno optimizira update-a se view sa prikazom rasporeda. Klikom na edit opciju poziva se metoda DozvoliEdit() nakon čeka je moguće ručno editovati raspored. Uzima se selektirani predmet unutar spinner-a ponuđenih predmeta i ističe na trenutnom rasporedu označavajući gdje se u sedmici nalaze termini sa tim predmetom. Nakon odgovarajućih izmjena i klika na save opciju poziva se metoda IzvršiIzmjenuZaPredmet koja ažurira trenutno stanje na rasporedu. Metoda spasiIzmjene() spašava dati raspored za studenta u bazu podataka.
LoginController ima metode validacijaKornisika() koja uporedi sa bazom podataka da li je trenutni input postojeći u bazi na osnovu username-a i MD5 hashiranog enkriptiranog koda za password. Ukoliko postoji izvršava se prijava i korisnik ulazi u sistem ukoliko ne postoji poziva se metoda InvalidLabel pomoću koje se ispisuje na ekranu poruka da username ili password nije ispravan.
AdministratorController posjeduje standardne edit, save i delete metode za upravljanje svim korisnicima, terminima, salama i predmetima unutar baze podataka.

9. Dijagram komponenti

Dijagram komponenti se sastoji od aktera koji koriste sistem što su student, profesor i administrator, interfejsa koji im se pružaju, komponenti koje realizuju zahtijevane interfejse – npr. optimizacija, obavijest, dodavanje termina; te interfejsa preko kojih komponente međusobno komuniciraju – npr. spašavanje promjena u bazu podataka, gdje su baza i uređivanje rasporeda komponente.

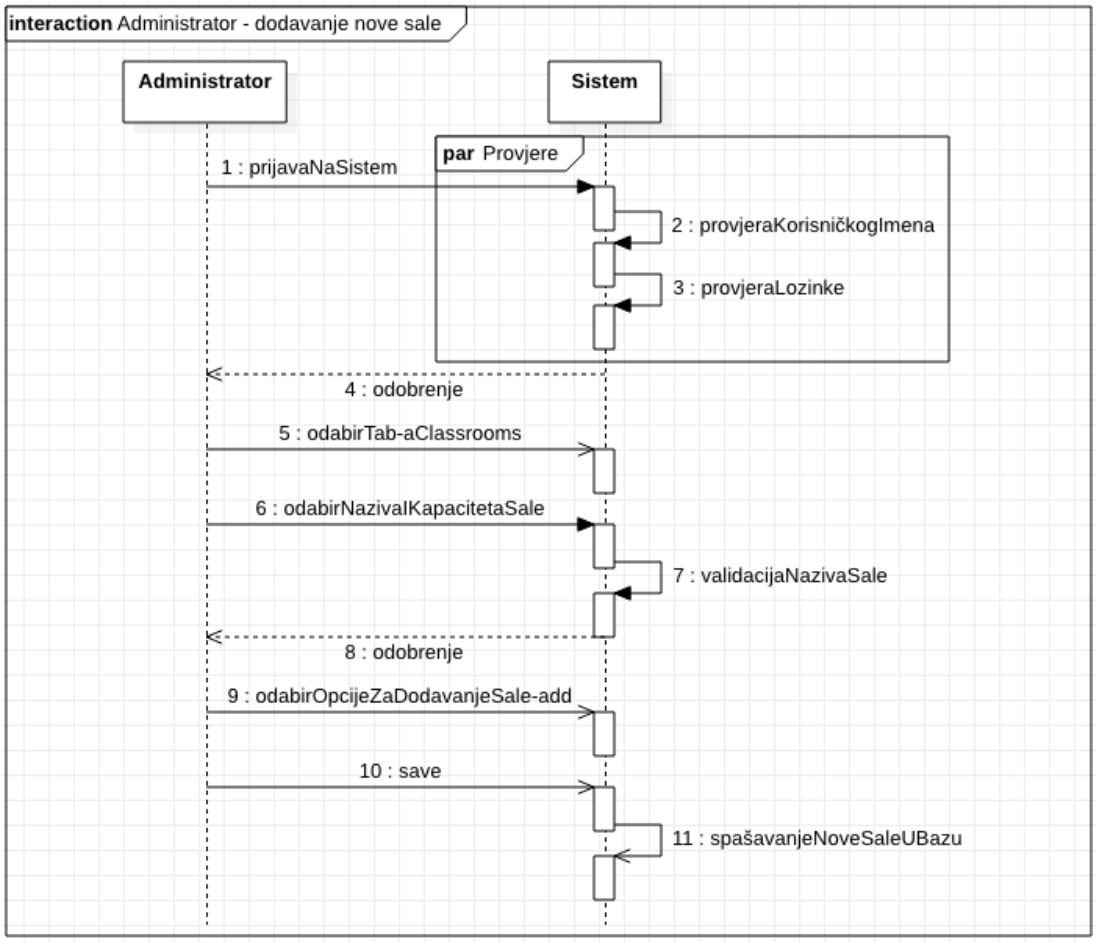


Slika 47: Dijagram komponenti

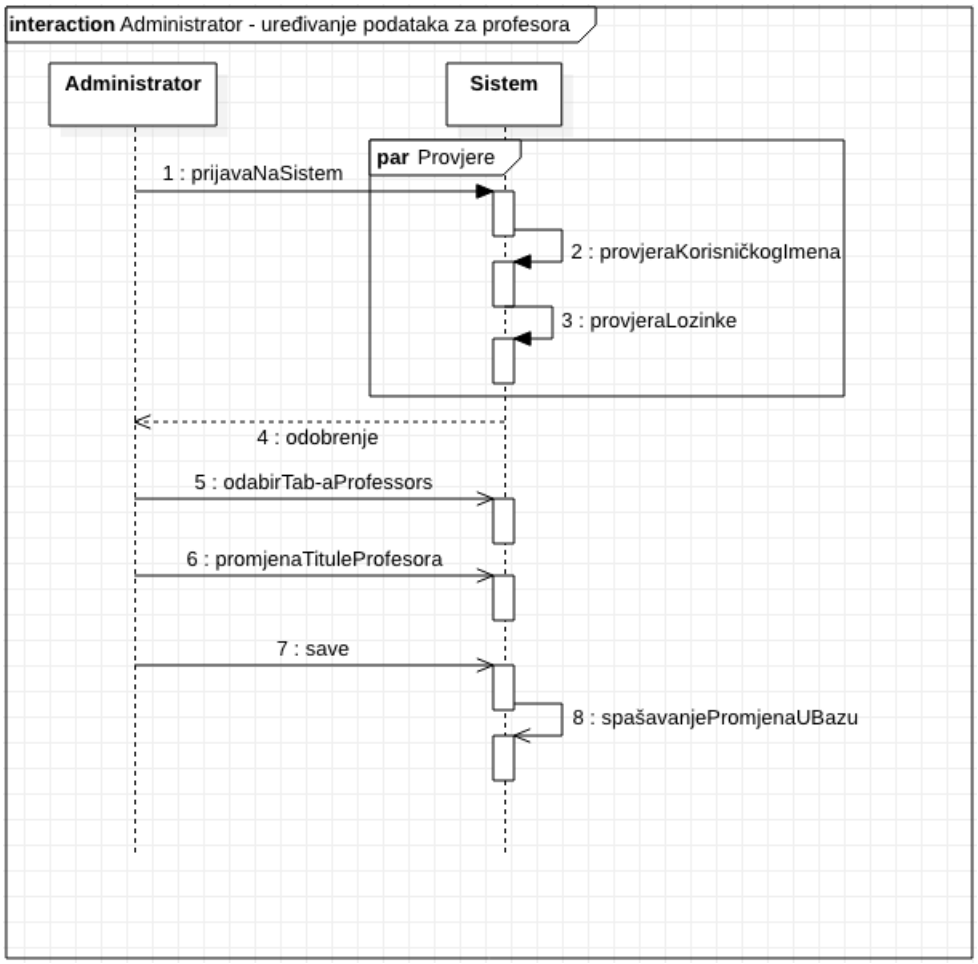
10. Dijagrami sekvence

Dijagrami sekvence podijeljeni su u tri skupine, dijagrami sekvence administratora, profesora i studenta. Prikazat ćemo neke od dijagrama sekvence u nastavku.

10.1 Dijagrami sekvence administratora



Slika 18: Administrator - Dodavanje nove sale

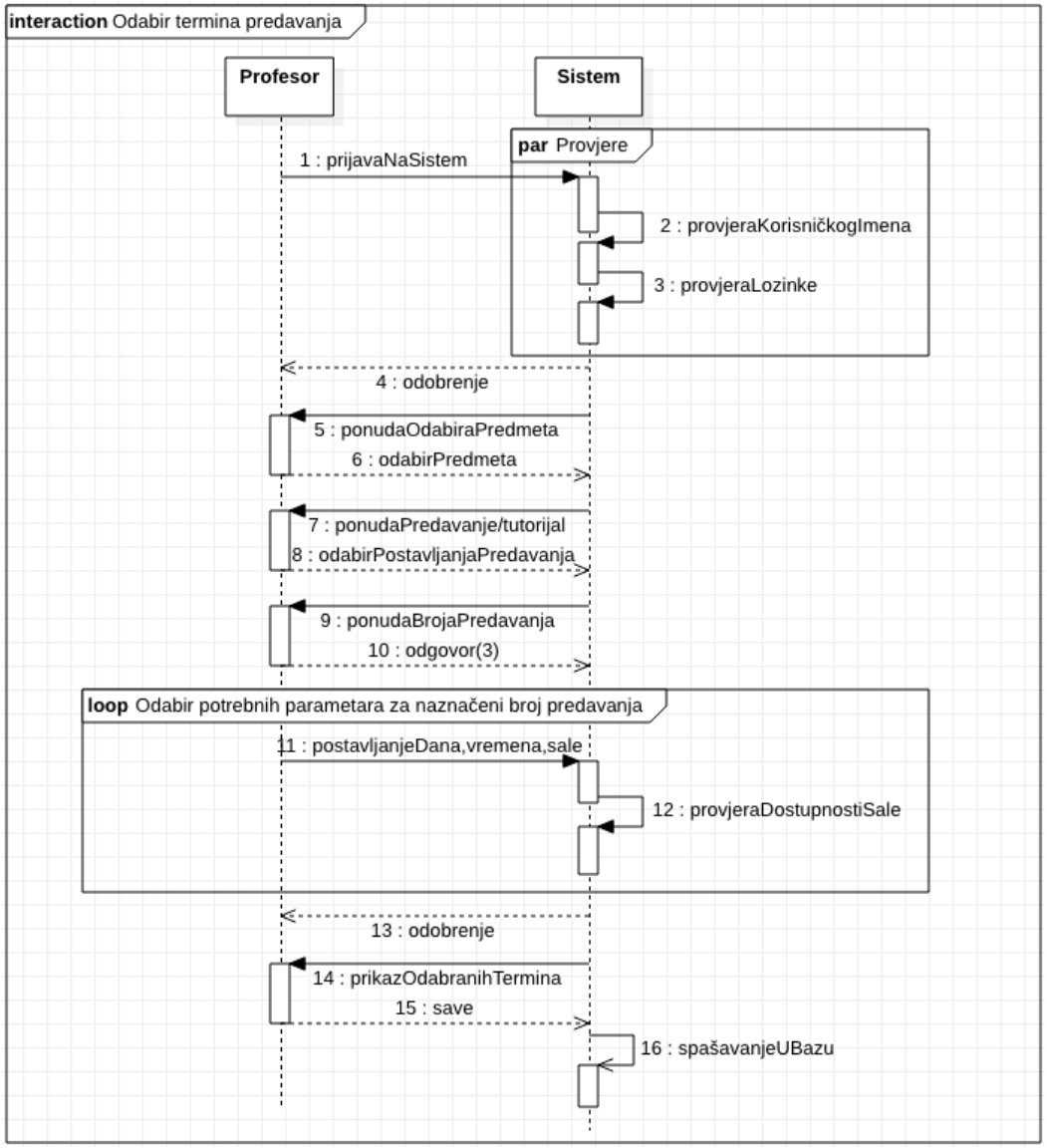


Slika 59: Administrator - Uređivanje podataka za profesora

10.2 Dijagrami sekvence profesora

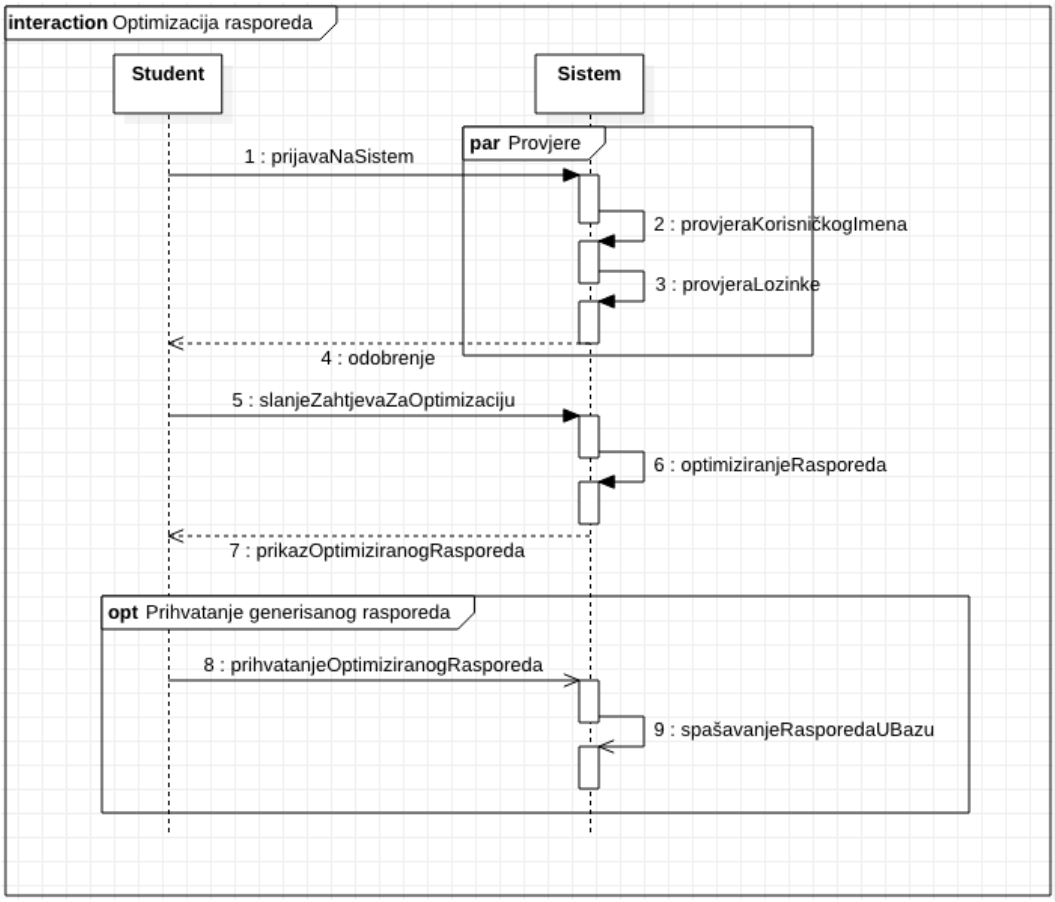
Dijagram sekvence prikazan na slici 20 zajednički je za administrator i profesora.

Kada se profesor prijavi na sistem sistem uporedo vrši provjeru koriničkog imena i lozinke nakon čega se profesoru u zavisnosti od tačnosti unesenih podataka odobrava ili odbija pristup aplikaciji. S obzirom na to da jedan profesor može predavati više predmeta, sistem nudi profesoru da izabere predmet za koji želi da doda termine, osim toga profesor još bira da li želi dodavati termine predavanja ili vježbi. U nastavku, profesor treba da bira broj, u ovom slučaju predavanja, koji mu je potreban u jednoj sedmici, iza čega možemo uočiti loop fragment koji naznačava da se radnja odabira dana, vremena i sale termina izvršava više puta, kao i validacija da li je željena sala u tom vremenu slobodna. Još na kraju, kao što ste i na formama vidjeli, profesoru se prikažu svi termini koje je odabrao i potom spašavaju u bazu.

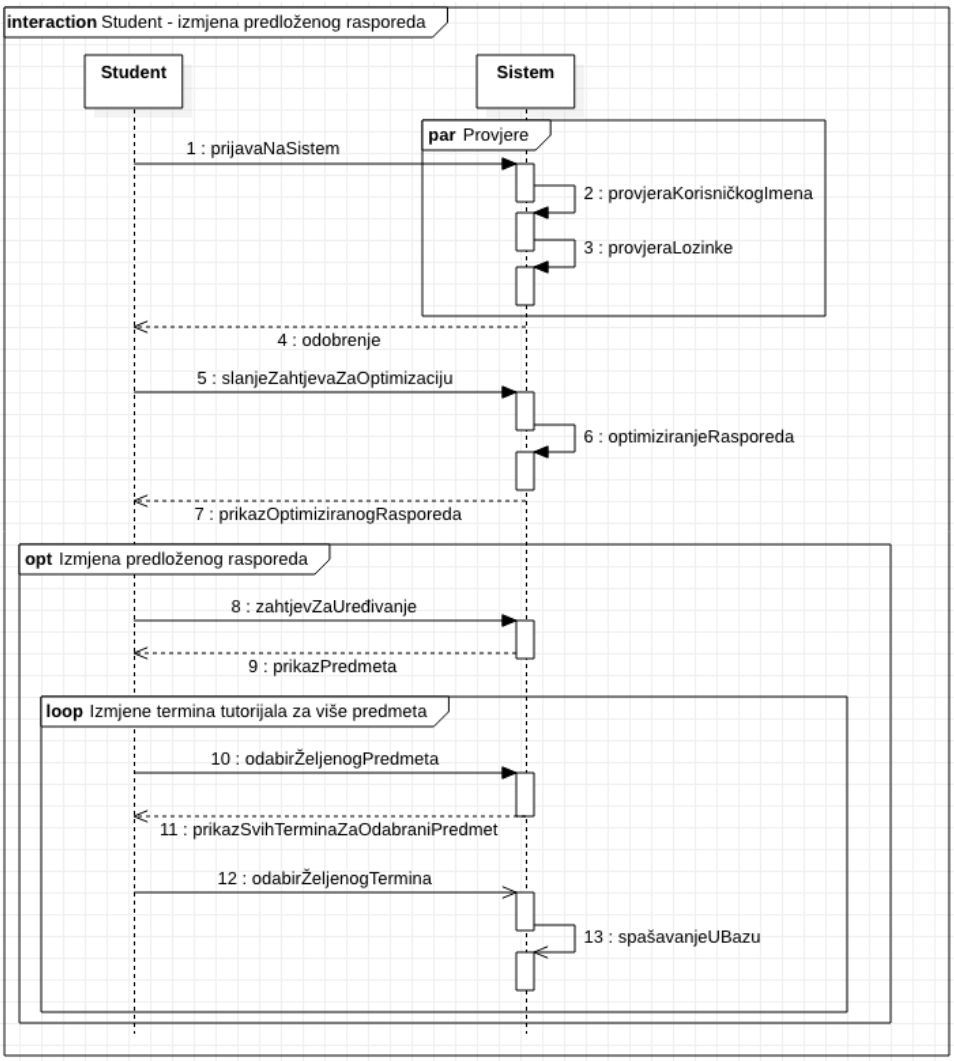


Slika 60: Profesor - Odabir termina predavanja

10.3 Dijagrami sekvence sudenta



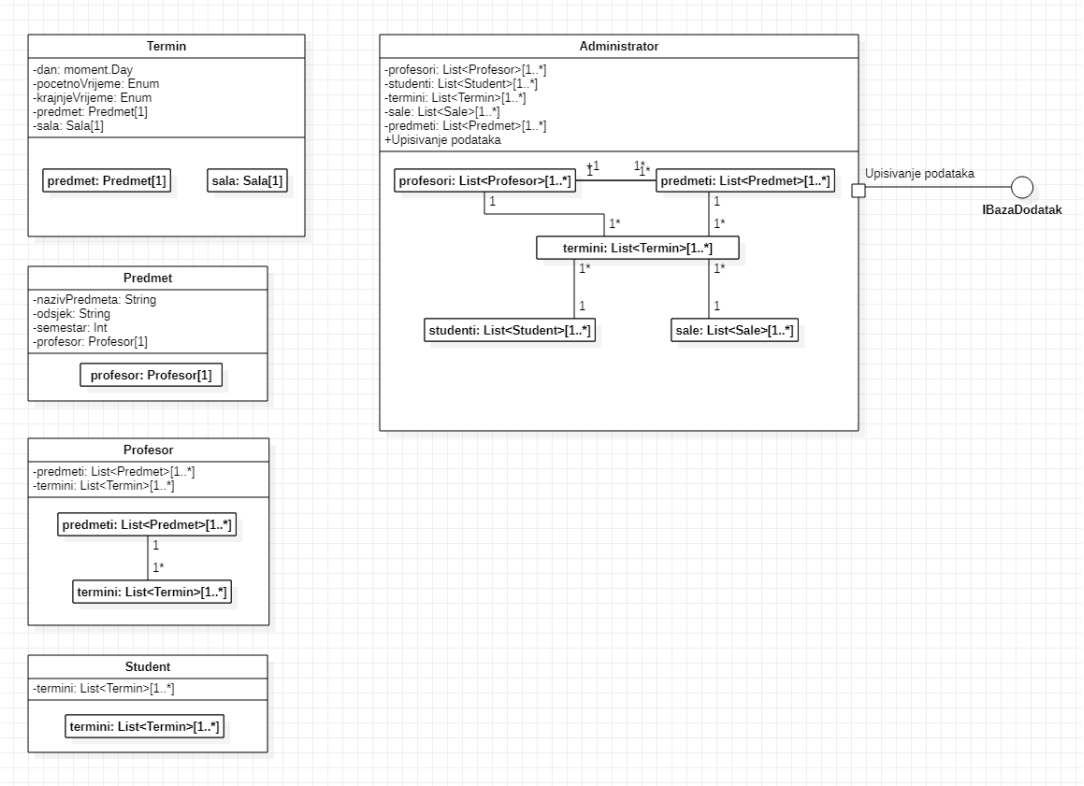
Slika 21: Student - Optimizacija rasporeda



Slika 22: Student - Izmjena predloženog rasporeda

11. Dijagrami složenih struktura

Dijagrami složenih struktura prikazuju interakcije elemenata sistema sa ostalim dijelovima. Razmotrit ćemo dijagram administratora, obzirom da su u toj klasi prisutne i sve ostale klase. Unutar dijagrama vidimo veze, npr. atribut sale tipa List<Sala> interno je vezan sa atributom termini koji je tipa List<Termin> vezom 1 naprema više obzirom da će jedna sala biti iskorištena za više termina. Svakako je prisutna i veza, tj. port, sa interfejsom Baza Dodatak obzirom da administrator vrši većinu manipulacija nad bazom.



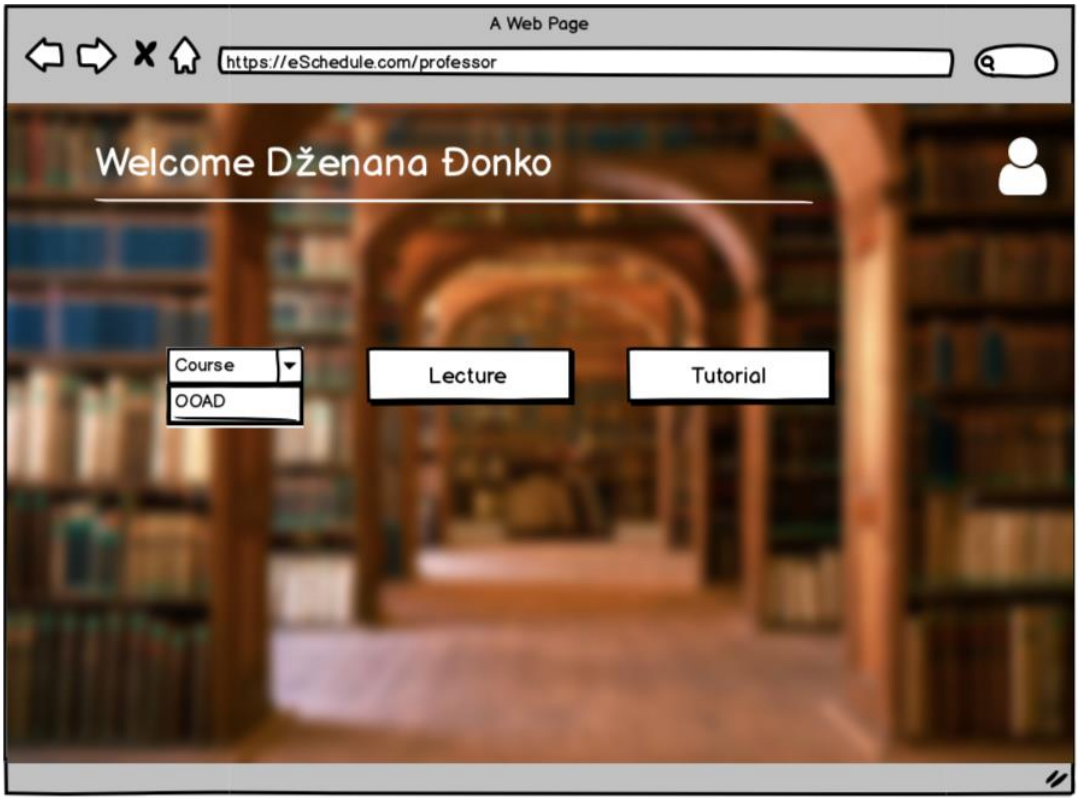
Slika 23: Dijagrami složenih struktura

12. Forme aplikacije

Početni zaslon, pri pristupu korisnika aplikaciji, izgleda kao na slici 24.

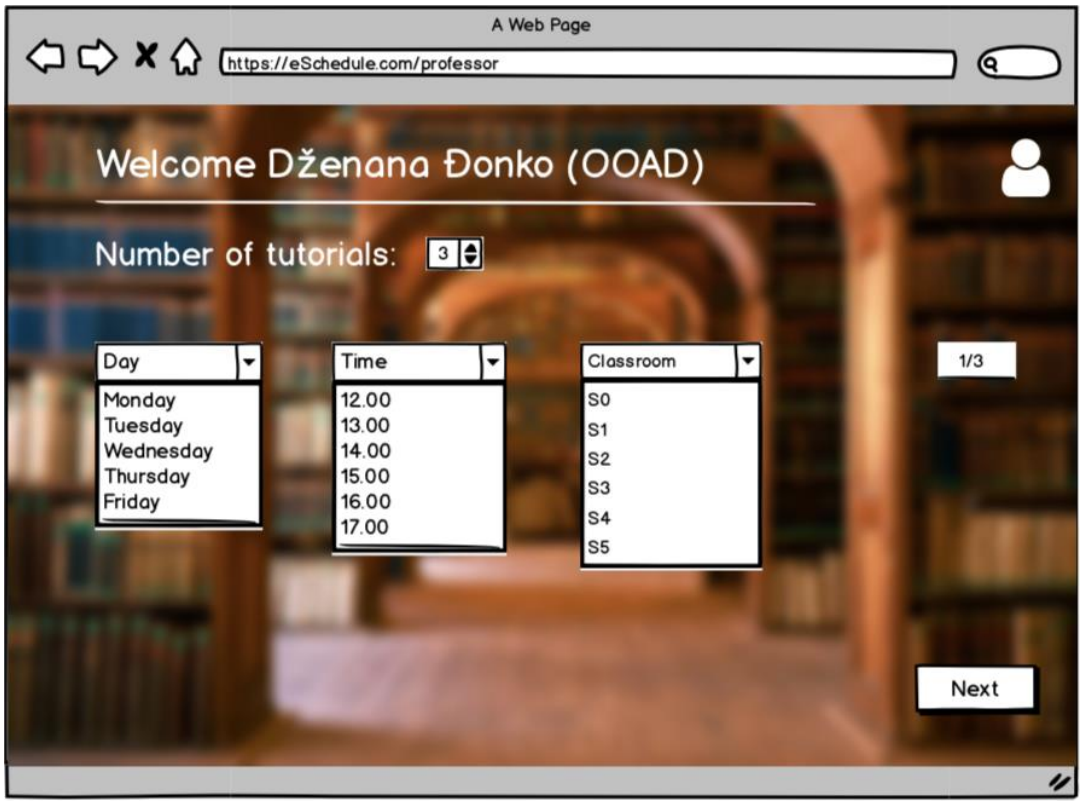
Slika 24: Početni zaslon

Ukoliko se prepozna da se professor prijavio na aplikaciju iduća forma će biti kao na slici 25. Ponuđen je odabir predmeta, ukoliko određeni profesor predaje više predmeta, te odabir između kreiranja termina za predavanja i tutorijale.

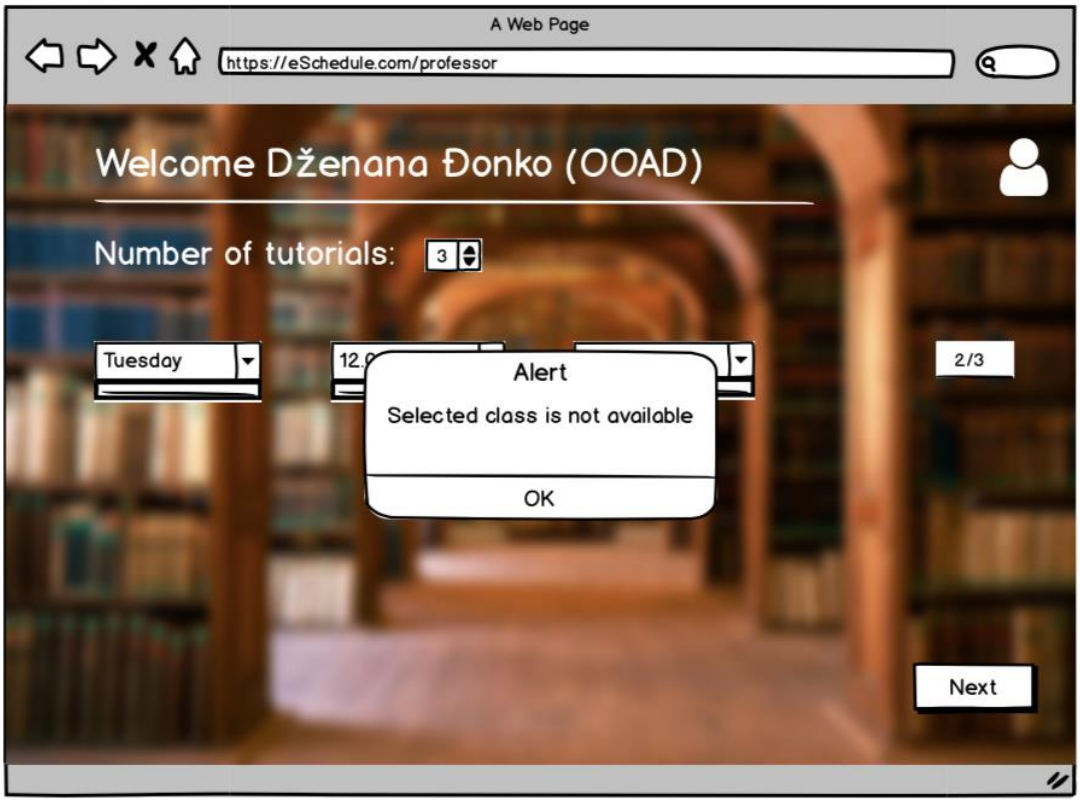


Slika 25. Početni zaslon profesora

Profesor će pri kreiranju odabrati koliko termina predavanja/tutorijača je potrebno. Ukoliko profesor pri kreiranju termina odabere takav termin da je sala u odabrano vrijeme zauzeta bit će ispisana poruka.

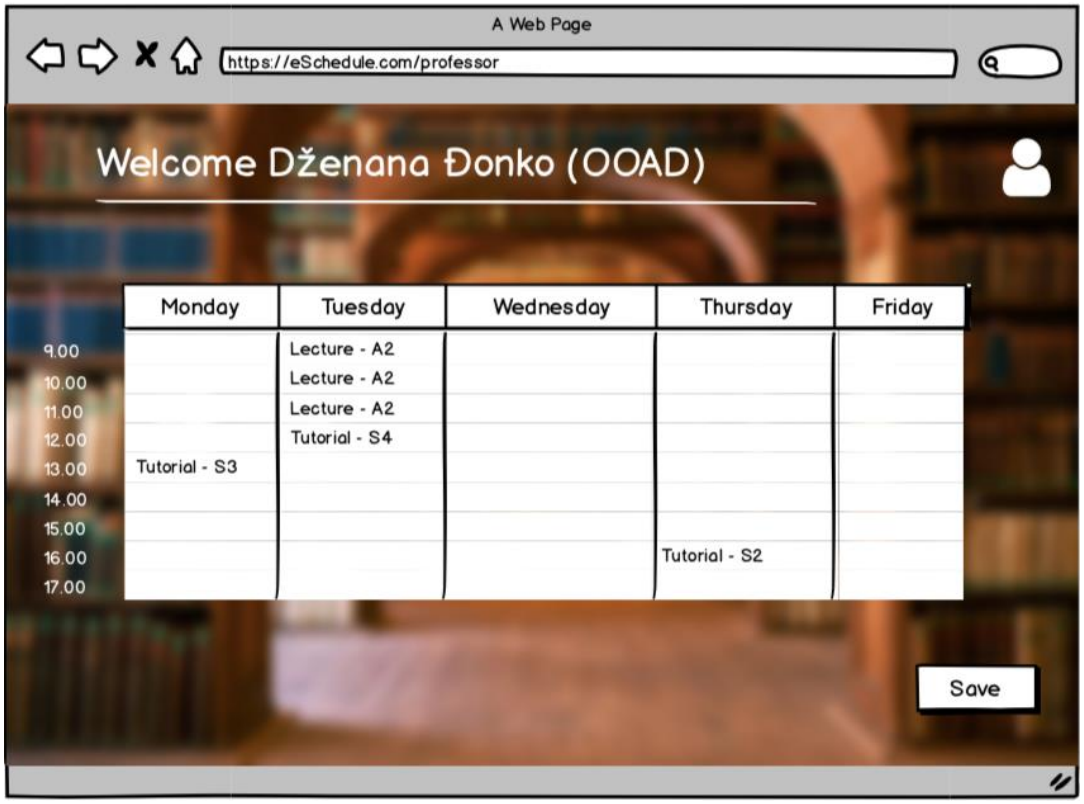


Slika 26: Kreiranje termina tutorijala



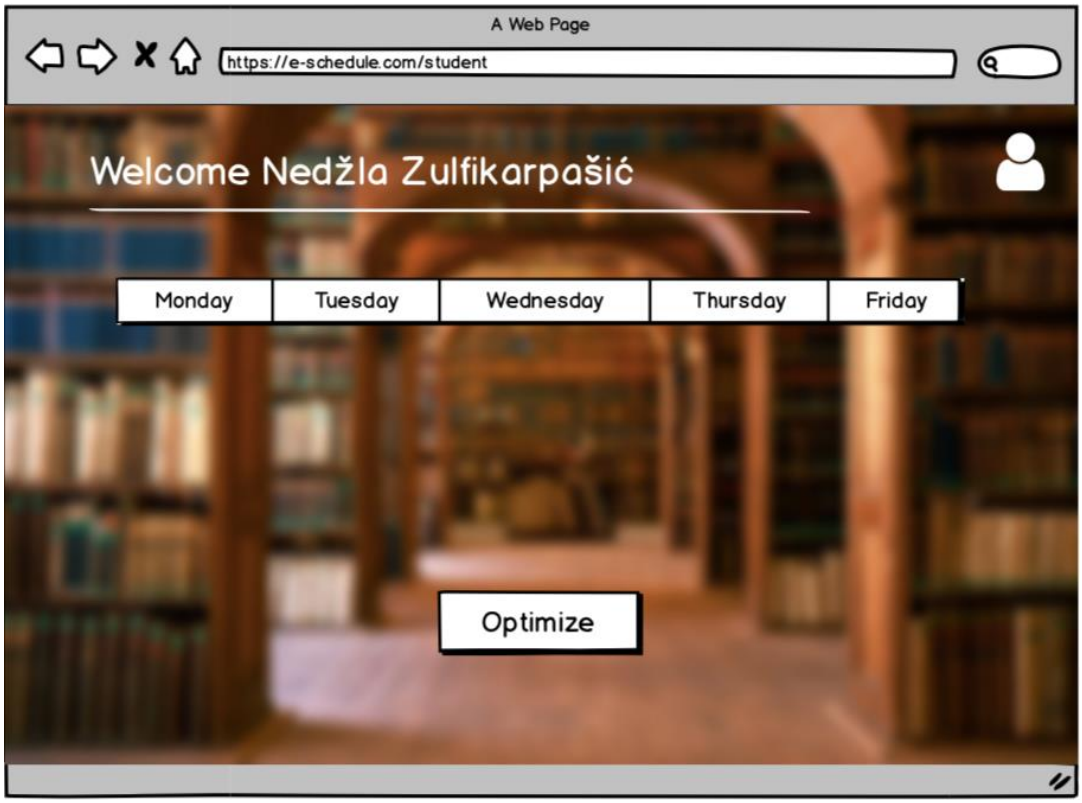
Slika 27: Odabrana sala je zauzeta

Po završetku kreiranja svih termina profesoru će biti ispisan raspored svih termina koje je kreirao.

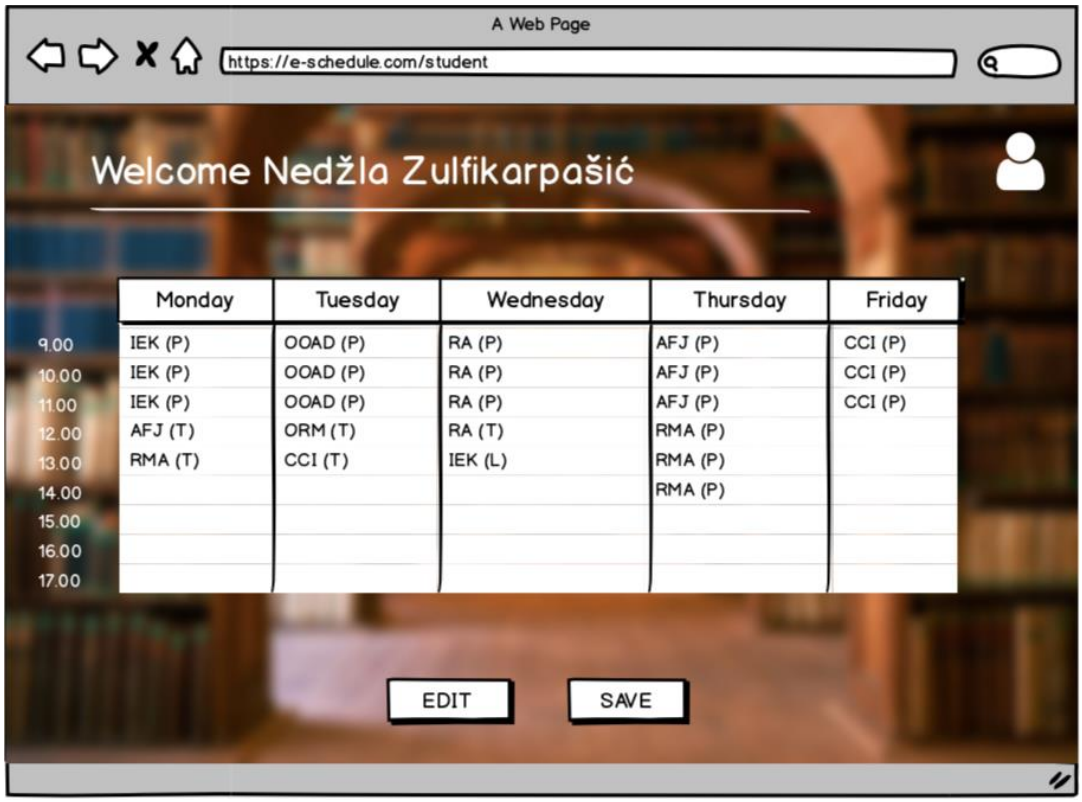


Slika 28: Raspored kreiranih termina

Ukoliko se na prijavi prepozna da je student pristupio aplikaciji, njegov prvi zaslon će izgledati kao na slici 29. Klikom na dugme “Optimize” ispisat će se najbolji raspoloživi raspored za određenog studenta.



Slika 29. Početni zaslon studenta

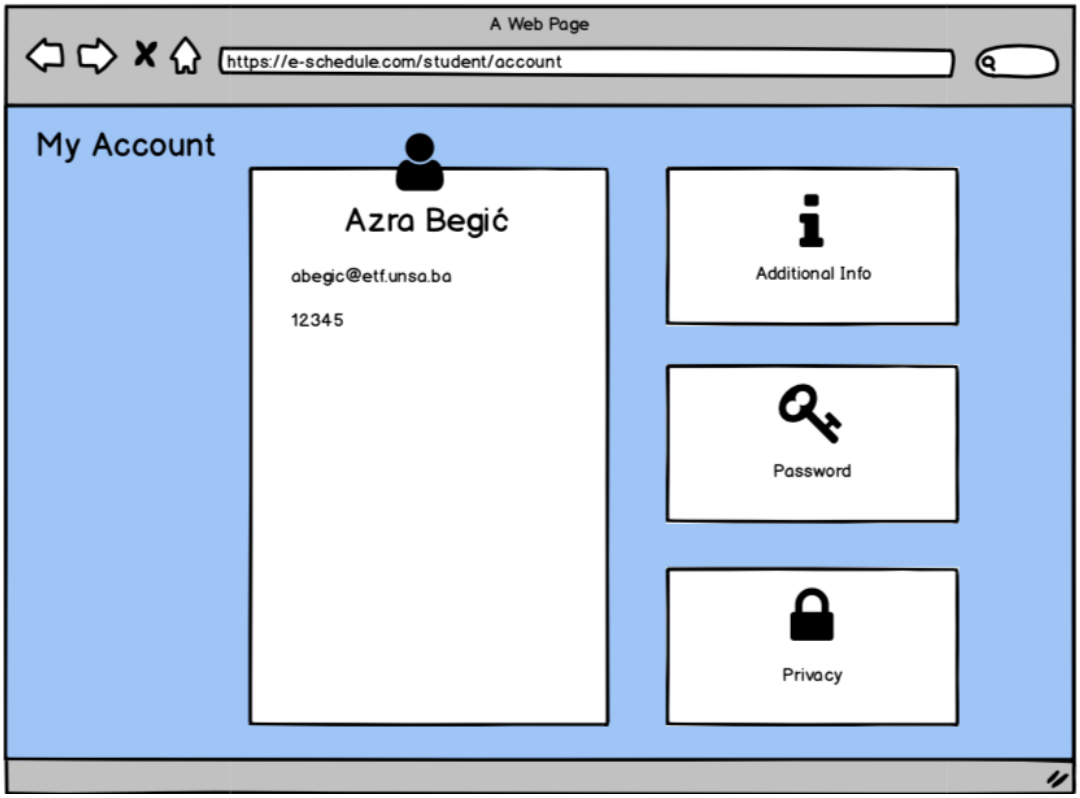


Slika 30: Optimizirani raspored

Ukoliko se student koji je prethodno generisao svoj raspored i prihvatio ga ponovo prijavi na aplikaciju bit će ispisana poruka kao na slici 31.

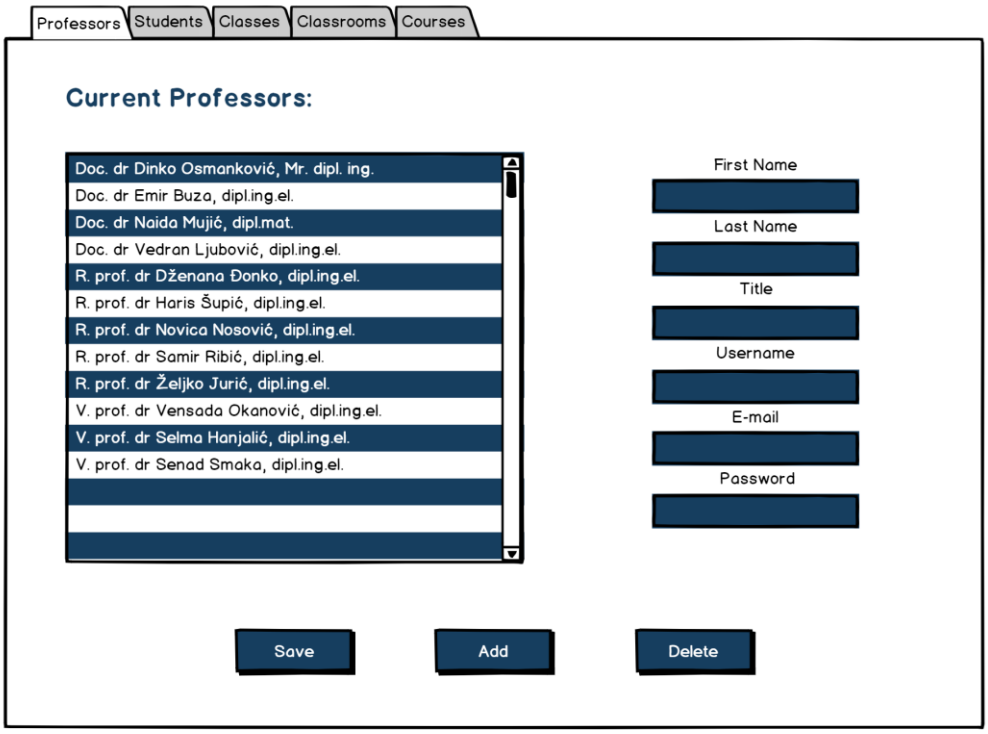
Slika 31: Raspored pretodno generisan i prihvaćen

Svim korisnicima će također biti ponuđena izmjena korisničkog profila.

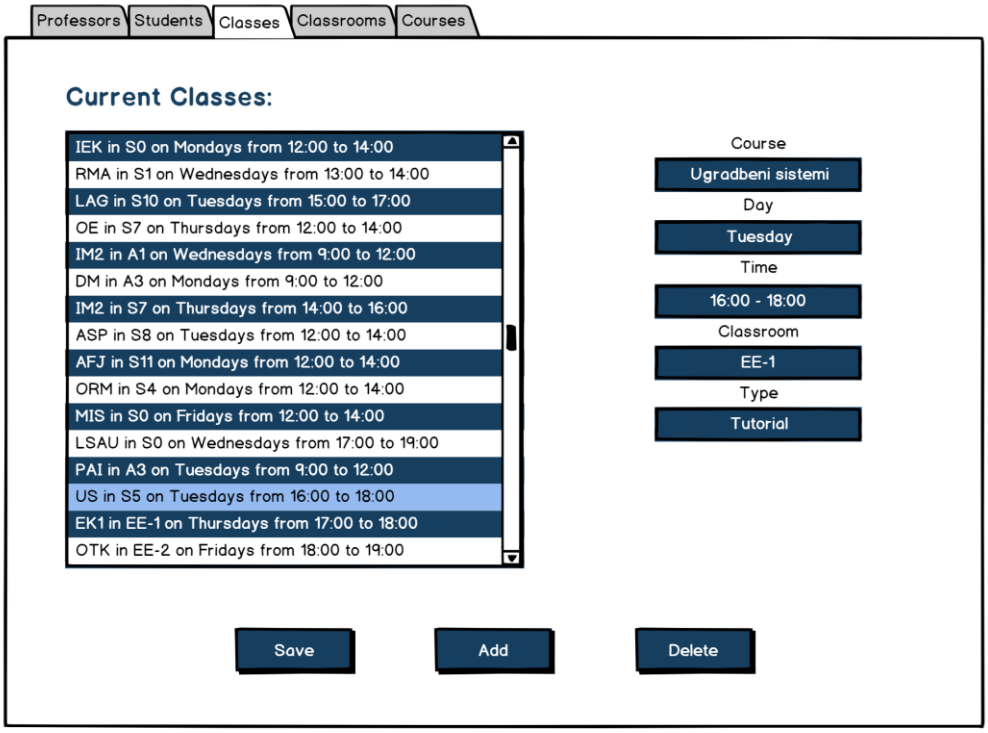


Slika 32: Izmjena korisničkih podataka

Ukoliko administrator pristupa aplikaciji prikazat će mu se zaslon sa prozorima za pregled profesora, studenata, svih termina, sala i predmeta. Administrator će biti u mogućnosti dodavanja, promjene i brisanja svih postojećih objekata u bazi podataka.



Slika 33: Administratorov pogled na sistem



Slika 34: Administratorov pogled na sistem

13. Zašto izabrati ovu aplikaciju i kako je u budućnosti unaprijediti?

Motiv za pravljenje aplikacije koja optimizira studentski raspored nastala je zbog svjesnosti da u našem rasporedu ima mnogo "neiskorištenog" vremena ili preklapanja u terminima tutorijala s obzirom na to da nam svi termini tutorijala ili vježbi nisu poznati u isto vrijeme. Na ovaj način, studentski raspored se i ne može formirati dok svi odgovarajući termini ne budu kreirani.

Aplikacija nudi i mogućnost izmjene termina tutorijala koji nam ne odgovaraju zbog ostalih ličnih obaveza.

Također, olakšava rad profesorima jer na ovaj način mogu jednostavno kreirati termine svojih predavanja i tutorijala/vježbi, bez da se moraju konsultovati sa kolegama kada je određena sala slobodna i slično. Na ovaj način sve potrebne provjere o validnosti kreiranog termina jako brzo obavlja sistem.

U svrhu pružanja usluge optimiziranja rasporeda svim studentima, nakon što svi profesori unesu potrebne termine, studentima će biti poslan mail obavještenja.

Što se tiče same optimizacije, korišteni algoritam garantuje najbolji mogući raspored uzimajući u obzir vrijeme kada je student pristupio aplikaciji. U narednom periodu, aplikacija se može unaprijediti tako što se doda nekoliko različitih alogitama u pogledu efikasnosti u odnosu na ukupan broj tutorijala koji su definisani od strane profesora za određeni semestar. Nakon što student zatraži optimizaciju, sistem bi mogao sam, na osnovu ukupnog broja unesenih tutorijala, odrediti koji algoritam da koristi.

Osim navedenog, u budućnosti može biti korisno da se dopusti i asistentima dodavnje termina tutorijala. S obzirom na to da na većini predmeta asistenti drže vježbe kojima možda ne bi odgovarali termini tutorijala koje su profesori izabrali, bilo bi korisno, ukoliko profesor odluči da asistent sam izabere vrijeme tutorijala, da se takva vrsta ovlaštenja i omogući.