

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. xxxx

Baza podataka i web-aplikacija za udrugu

Karlo Sudec

Zagreb, lipanj 2020.

Ova stranica treba biti:

ILI prazna stranica

ILI stranica sa zahvalom (po želji studenta; zahvala nije obvezna).

Sadržaj

Uvod	1
1. Specifikacija zahtjeva	2
1.1. Domena problema.....	2
1.2. Analiza zahtjeva	3
2. Baza podataka.....	7
2.1. ER model baze podataka	7
2.1.1. Dijagram ER modela baze podataka	8
2.1.2. Entiteti	9
2.1.3. Veze	10
2.2. Relacijski model baze podataka	13
2.2.1. Detaljan opis relacijskog modela.....	13
2.3. Integritet baze podataka.....	17
2.4. Implementacija baze podataka.....	27
2.4.1. Implementacija relacija.....	27
2.4.2. Implementacija okidača i procedura.....	34
2.5. Podatci za testiranje i daljnji razvoj.....	36
3. Web aplikacija	37
3.1. Arhitektura web aplikacije.....	37
3.1.1. Implementacija upravljača.....	38
3.1.2. Implementacija modela.....	40
3.1.3. Implementacija pogleda.....	43
3.2. Korištene tehnologije.....	45
4. Korisničke upute.....	50
4.1. Autorizacija korisnika.....	50
4.2. Student.....	52

4.2.1.	Članovi	52
4.2.2.	Timovi	53
4.2.3.	Okupljanja	54
4.2.4.	Internacionalni događaji	55
4.2.5.	Lokalni ogranci udruge.....	57
4.2.6.	Projekti	58
4.3.	Član upravnog odbora	59
4.4.	Administrator.....	61
Zaključak		62
Literatura		63
Sažetak.....		64
Summary.....		65

Uvod

Vođenje bilo koje organizacije zahtjeva prikupljanje, organizaciju, čuvanje i pristup relativno velikoj količini podataka. Udruge u Republici Hrvatskoj su zakonski obavezne voditi registar članova s propisanim minimalnim podacima koji se prikupljaju. Također, većina udruga prikuplja i čuva dodatne podatke o financijama, projektima koje organiziraju, sastancima, partnerima s kojima surađuju, natjecajima...

Ovaj rad će se usmjeriti na studentsku udrugu i izradu programske potpore koja može pomoći u upravljanju udrugom i podacima koje ona obrađuje. Glavni dijelovi programske potpore su relacijska baza podataka i web aplikacija.

Baza podataka je dizajnirana je tako da zadovoljava trenutne korisničke zahtjeve i da omogućiti daljnje nadogradnje same baze i programa koji ju koriste.

Web aplikacija služi za prikaz i izmjenu podataka u bazi, a namijenjena je članovima udruge, članovima upravnog odbora i administratorima. Upravni odbor može pristupiti svim podacima te ih može uređivati i dodavati nove. Članovi mogu pregledavati aktivnosti udruge. Cilj je da korištenje aplikacije bude intuitivno i jednostavno.

Za izradu baze podataka koristi se PostgreSQL, dok su u web aplikaciji korišteni programski jezik Java, Spring framework, Hibernate framework, HTML5 i CSS.

U sljedećim poglavljima opisan je problem koji se pokušava riješiti, modelirana je i implementirana baza podataka, oblikovana je i implementirana web aplikacija. Na kraju se nalaze korisničke upute za web aplikaciju.

1. Specifikacija zahtjeva

1.1. Domena problema

U radu se pokušava riješiti problem prikupljanja, čuvanja i dohvaćanja podataka za vođenje studentske udruge. Prikupljaju se osnovni podaci o studentima koji dođu na neko od okupljanja udruge, prijave se za sudjelovanje na nekom od projekata udruge ili ispune online obrazac. Osnovni podaci služe uglavnom isključivo za kontaktiranje studenata i informiranje o aktivnostima udruge. Kada student dostavi podatke koji su propisani Zakonom o udrugama, upisuje se u registar članova te se bilježi da je postao član udruge. Prema Pravilniku udruge određena su pravila za status “aktivnog člana” udruge. Aktivni članovi imaju pravo glasa na sjednicama Skupštine. Potrebno je voditi evidenciju aktivnih članova.

U sklopu udruge postoje timovi koji se bave raznim dijelovima važnim za udugu (HR tim, PR tim, IT tim, tim za prikupljanje novčanih sredstava...). Svaki tim ima voditelja kojeg Upravni odbor bira na mandat od jedne godine (akademske). Članovi tima su studenti koji se dobrovoljno jave za sudjelovanje u timu.

Studentska udruga organizira razne projekte tijekom godine. Projekti su uglavnom stručne edukacijske radionice za studente. Upravni odbor izabire organizacijski odbor za svaki projekt. Organizatori su studenti, članovi udruge. Za sudjelovanje na projektu se mogu prijaviti svi studenti.

Partneri udruge (pravne osobe) sponzoriraju projekte udruge nekim od sponzorskih paketa. Sponzorski paketi sadrže stavke koje se udruga obvezuje izvršiti ukoliko partner sponzorira projekt. Jedan projekt može sponzorirati više partnera.

Redovno se održavaju okupljanja za članove (i zainteresirane studente). Potrebno je voditi evidenciju koji su studenti prisustvovali kojim okupljanjima. Okupljanja mogu biti formalna (sastanci, sjednice Skupštine...) i neformalna (druženja, team building, zabave...). Timovi također održavaju okupljanja za članove tima.

Udruga ima i međunarodnu razinu. U Europi postoje ogranci udruge na raznim sveučilištima. Svi lokalni ogranci organiziraju međunarodne događaje na koje se studenti

mogu prijaviti. Moguće je da više lokalnih ogranaka zajedno organizira jedan događaj. Događaji se dijele na 4 kategorije: edukacijske radionice, razmjene studenata, operacijski događaji i motivacijski vikendi. Edukacijske radionice u pravilu traju tjedan dana i imaju obavezan akademski dio s unaprijed određenom temom. Temu određuju organizatori. Razmjene studenata također traju tjedan dana. One nemaju akademskog dijela već je naglasak na druženju i umrežavanju studenata, međunarodnoj suradnji te upoznavanju kulture i običaja države domaćina. Operacijski događaji imaju za cilj unaprijediti one vještine studenata koje su temeljne za vođenje udruge (upravljanje projektima, rad u timu, organizacija vremena, emocionalna inteligencija...). Motivacijski vikendi dolaze nakon svakog od prethodno opisanih događaja, a na njima sudjeluje velik broj studenata iz okolice domaćina. Potrebno je voditi evidenciju o lokalnim ograncima i događajima te studentima koji su putovali na te događaje.

Važno je evidentirati i financijsko stanje udruge. Vodi se evidencija računa. Računi mogu biti izlazni (predstavljaju prihode) i ulazni (predstavljaju rashode). Računi mogu biti vezani za neku od aktivnosti udruge (projekt ili okupljanje).

1.2. Analiza zahtjeva

Iz domene problema očito je da postoje barem dvije vrste korisnika programske potpore. To su studenti i članovi upravnog odbora udruge.

Studenti moraju imati mogućnost prijave u sustav te pregleda osnovnih podataka o radu udruge, tj. uvid u opće podatke iz svih područja djelovanja udruge opisanima u domeni problema osim financija i osobnih podataka članova. Članovi upravnog odbora mogu pregledavati, izmjenjivati i uklanjati sve podatke. Također, zbog praktičnosti, dodat će ulogu administratora koji će upravljati dozvolama korisničkih računa svih korisnika.

U tablici 1. nalazi se pregled dozvola ovisno o ulogama korisničkog računa aplikacije.

Tablica 1. Pregled dozvola za pristup podacima ovisno o ulozi korisničkog računa

VRSTA PODATKA	STUDENT	UPRAVNI ODBOR
Članovi udruge	pregled (samo osnovni podatci)	pregled i izmjena
Timovi	pregled naziva i voditelja	pregled i izmjena naziva, voditelja i članova
Okupljanja	pregled	pregled i izmjena
Lokalni ogranci	pregled	pregled
Međunarodni događaji	pregled	pregled i izmjena
Projekti	pregled (samo osnovni podatci)	pregled i izmjena
Financije	/	pregled i izmjena

Potrebno je omogućiti registraciju u sustav kako bi zainteresiranim studentima bilo što lakše uključiti se u rad udruge. Svaki student mora imati korisničko ime i lozinku kako bi se mogao prijaviti u sustav.

Za studente je najvažnije evidentirati podatke za kontakt:

- Ime i prezime
- Email adresa
- Broj mobitela

Ukoliko je student član udruge, potrebno je evidentirati i:

- OIB
- Datum rođenja
- Mjesto prebivališta

Za timove je potrebno evidentirati:

- Naziv
- Opis
- Voditelj
- Članovi

Za okupljanja je potrebno evidentirati:

- Vrijeme
- Vrsta
- Prisutni studenti

Za lokalne ogranke je potrebno evidentirati:

- Naziv (naziv će ujedno biti i mjesto u kojem se nalazi)
- Država
- Događaji koje su organizirali

Za međunarodne događaje je potrebno evidentirati:

- Naziv
- Vrsta
- Vrijeme
- Prisutni studenti
- Država u kojoj se nalazi

Za projekte je potrebno evidentirati:

- Naziv
- Vrijeme
- Sudionici
- Partneri i sponzorski paketi

Za financije je potrebno evidentirati račune s podacima:

- Iznos
- Vrsta
- Vrijeme
- Povezanost s projektom/okupljanjem

Iz navedenog vidim da je povezanost između raznih entiteta važan dio zahtjeva. Potrebno je izraditi korisničko sučelje koje će intuitivno pokazivati kako su entiteti povezani.

2. Baza podataka

Iz opisa problema i analize zahtjeva je vidljivo da će programska potpora morati spremati i obrađivati relativno velike količine podataka. Za to će služiti baza podataka.

Baza podataka je skup međusobno povezanih podataka, pohranjenih u vanjskoj memoriji računala. Podaci su istovremeno dostupni raznim korisnicima i aplikacijskim programima.[1] Rad s podacima u bazi omogućuje program koji nazivamo *sustav za upravljanje bazom podataka* (eng. *DBMS*). Sustav za upravljanje bazom podataka predstavlja dodatan „sloj“ nad fizički pohranjenim podacima. On određuje kako su podatci fizički spremljeni u memoriji u skladu s zadanom logičkom strukturom. Sustav izvršava sve izmjene u bazi te osigurava integritet i sigurnost podataka u bazi. Također, pomoću sustava izvršavamo upite nad bazom. U ovom radu koristi se PostgreSQL sustav za upravljanje bazom podataka.

Svaka baza podataka je modelirana u skladu s modelom. *Model podataka je skup pravila koja određuju kako sve može izgledati logička struktura baze podataka. Model čini osnovu za oblikovanje i implementiranje baze.*[1] Kroz povijest je razvijeno više modela. U ovome radu koristi se relacijski model podataka. Relacijski model podataka zasniva se na pojmu relacije. Entiteti i veze među njima se prikazuju tablicama. Redak tablice predstavlja jedinku entiteta (ili veze), dok stupci tablice predstavljaju vrijednosti atributa tog entiteta za svaku jedinku.

2.1. ER model baze podataka

ER model (eng. *Entity–relationship model*) podataka je apstraktna organizacija podataka koji će biti u bazi. On se sastoji od entiteta i veza između njih. *Entitet je nešto što postoji u stvarnosti ili u svijesti; predmet promatran kao pojava s materijalnim jedinstvom, ali kojemu se objektivno postojanje temelji samo na odnosima.* [8]

Sve jedinke entiteta imaju attribute koji ih jednoznačno određuju i pobliže opisuju. Atributi koji jednoznačno određuju jedinke entiteta nazivaju se ključevima. Ključ može biti primarni ili alternativni. Sustav za upravljanje bazom podataka koristi primarne ključeve za sve akcije nad jedinkama tj. podacima. Svaki entitet mora imati primarni ključ.

Alternativni ključ označava da se taj skup atributa može koristiti za identifikaciju jedinki i često se prema njemu, u praksi, pretražuju jedinice entiteta.

Postoji više vrsta veza između entiteta, a razlikujemo i njihove kratnosti. Veze mogu imati i vlastite attribute. U podjela veza po broju entiteta koje povezuju najčešće su:

- unarne – povezuju entitet sa samim sobom (npr. student daje instrukcije studentu)
- binarne – povezuju dva entiteta (npr. student polaže ispit)
- ternarne – povezuju tri entiteta (npr. student polaže ispit na ljetnom roku 2020.)

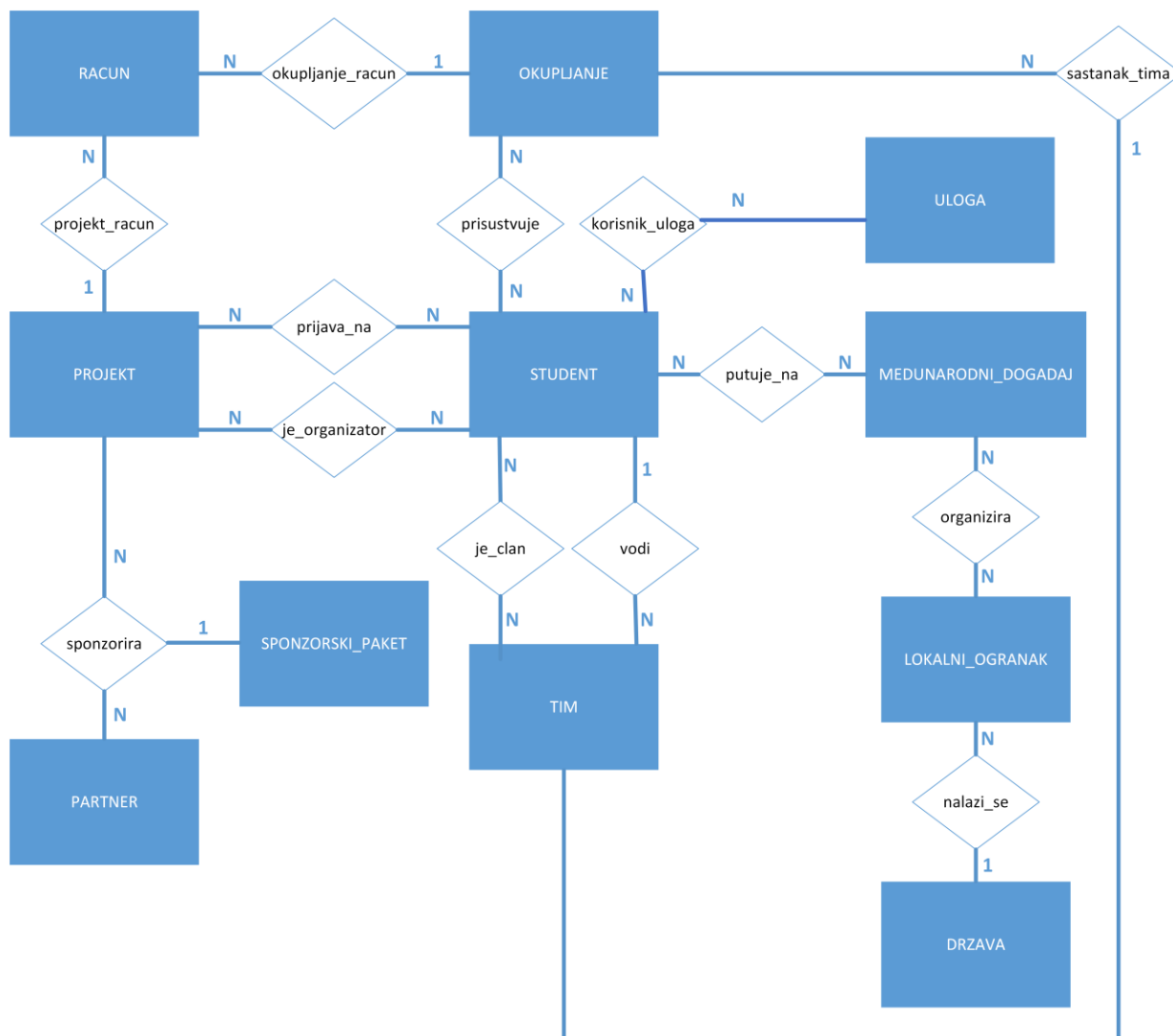
Kratnost veza između entiteta označava koliko jedinki iz entiteta može ostvariti tu vezu sa koliko jedinki iz drugog entiteta. U označavanju kratnosti oznaka *N* u pravilu označava niti jednu ili više jedinki, dok oznaka *1* označava točno jednu jedinku. Binarne veze mogu biti sljedećih kratnosti:

- 1:1
- 1:N
- N:N

U ovom radu se koristi upravo ER model jer se lako preslikava u relacijski model podataka i jednostavno se grafički oblikuje iz specifikacije zahtjeva.

2.1.1. Dijagram ER modela baze podataka

Na slici 1. prikazan je dijagram ER modela dobiven iz specifikacije zahtjeva (poglavlje 1.). Zbog preglednosti su izostavljeni atributi. Detaljan opis entiteta i veza s atributima i ključevima slijedi u poglavljima 2.1.2. i 2.1.3. Entitet „ULOGA“ služi kako bi svakom korisničkom računu bile dodijeljene odgovarajuće ovlasti (student, upravni odbor, administrator).



Slika 1. Dijagram ER modela baze podataka

2.1.2. Entiteti

Slijedi popis svih entiteta s njihovim atributima. Primarni ključevi su podcrtani i podebljani. Oznaka AK predstavlja da su ti atributi alternativni ključevi.

- STUDENT – **id_student**, ime_student, prezime_student, mail_student, mobitel_student, oib_student, datum_rodenja_student, prebivaliste_student, fakultet_student, godina_studija, smjer_studija, datum_azuriranja, je_clan, je_aktivan_clan, korisnicko_ime, lozinka

$AK1 = \{korisnicko_ime\}$, $AK2 = \{oib_student\}$

Pretpostavka je da se za studente podatak o godini studija svake godine ručno ažurira. Zato postoji atribut datum_azuriranja. Atributi korisnicko_ime i lozinka služe za prijavu u sustav.

- TIM – **id_tim**, naziv_tim, opis_tim
- PROJEKT – **id_projekt**, naziv_projekt, opis_projekt, pocetak_projekt, zavrsetak_projekt
- PARTNER – **id_partner**, naziv_partner, oib_partner, web_adresa_partner, mobitel_partner, mail_partner
 $AK = \{oib_partner\}$
- SPONZORSKI PAKET – **id_sponzorski_paket**, naziv_sponzorski_paket, stavke, datum_stvaranja
- OKUPLJANJE – **id_okupljanje**, naziv_okupljanje, opis_okupljanje, pocetak_okupljanje, zavrsetak_okupljanje, je_formalno
- MEDUNARODNI DOGADAJ – **id_medunarodni_dogadaj**, naziv_medunarodni_dogadaj, opis_medunarodni_dogadaj, pocetak_medunarodni_dogadaj, zavrsetak_medunarodni_dogadaj, vrsta_medunarodni_dogadaj, kapacitet, cijena
- LOKALNI OGRANAK – **id_lokalni_ogranak**, naziv_lokalni_ogranak, opis_lokalni_ogranak
- DRZAVA – **id_drzava**, naziv_drzava
- RACUN – **id_racun**, vrsta_racun, iznos_racun, vrijeme_racun, napomena
- ULOGA – **id_uloga**, naziv_uloga

2.1.3. Veze

Slijedi popis svih veza s atributima. Primarni ključevi su podcrtani i podebljani. Entiteti koje povezuju navedeni su u zagradama.

- prisustvuje (STUDENT - OKUPLJANJE) – **id_student, id_okupljanje**
Na jednom okupljanju je više studenata, a jedan student može biti na više okupljanja.
Veza je N:N.
- prijava_na (STUDENT - PROJEKT) – **id_student, id_projekt**, je_prihvacen
Jedan student se može prijaviti na više projekata, a na jednom projektu sudjeluje više studenata.
Veza je N:N.
- je_organizator (STUDENT - PROJEKT) – **id_student, id_projekt**
Jedan student može biti organizator na više projekata, a na jednom projektu postoji više organizatora.
Veza je N:N.
- putuje_na (STUDENT - MEDUNARODNI_DOGADAJ) –
id_student, id_medunarodni_dogadaj, je_prihvacen, napomena
Jedan student se može prijaviti za putovanje na više događaja, a za jedan događaj se može prijaviti više studenata.
Veza je N:N.
- vodi (STUDENT - TIM) - **id_tim**, id_student
Jedan student može voditi više timova, a jedan tim ima isključivo jednog voditelja.
Veza je 1:N.
- je_clan (STUDENT - TIM) – **id_student, id_tim**, napomena
Jedan student može biti član više timova, a u jednom timu se nalazi više studenata.
Veza je N:N.
- sastanak_tima (TIM - OKUPLJANJE) – **id_okupljanje**, id_tim
Jedan tim ima više sastanaka, a jedan sastanak je isključivo od jednog tima.
Veza je 1:N.

- sponzorira (PROJEKT - PARTNER - SPONZORSKI_PAKET) -

id projekt, id partner, id_sponzorski_paket, iznos, napomena

Jedan projekt jednim sponzorskim paketom može sponzorirati više članova.

Jedan partner može sponzorirati više projekata s istim sponzorskim planom.

Jedan projekt jedan partner sponzorira s jednim sponzorskim paketom.

Veza je 1:N:N.

- projekt_racun (PROJEKT - RACUN) – **id racun**, id_projekt

Za jedan projekt može biti izdano više računa. Jedan račun je isključivo za jedan projekt.

Veza je 1:N.

- okupljanje_racun (OKUPLJANJE - RACUN) – **id racun**, id_okupljanje

Za jedno okupljanje može biti izdano više računa. Jedan račun je isključivo za jedno okupljanje.

Veza je 1:N.

- organizira (MEDUNARODNI_DOGADAJ - LOKALNI_OGRANAK) –

id međunarodni događaj, id lokalni ogranak

Jedan lokalni ogranak organizira više međunarodnih događaja. Također, moguće je da jedan međunarodni događaj organizira više lokalnih ogranaka zajedno.

Veza je N:N.

- nalazi_se (LOKALNI_OGRANAK - DRZAVA) –

id lokalni ogranak, id_drzava

U jednoj državi može postojati više lokalnih ogranaka.

Veza je 1:N.

- korisnik_uloga (STUDENT - ULOGA) –

id student, id uloga

Jedan student može imati više uloga. Jednu ulogu može imati više studenata.

Veza je N:N.

2.2. Relacijski model baze podataka

Relacijski model se može lako dobiti iz ER modela opisanog u poglavlju 2.1. Svaki entitet će biti jedna relacija. Također, svaka veza iz ER modela postaje relacija. Ukoliko su primarni ključevi u tako dobivenim relacijama jednaki, te dvije relacije se pripajaju u jednu. Atributi se nadodaju jedni drugima.

Ovdje se uvodi pojam strani ključ. Strani ključ je skup atributa u jednoj relaciji koji pokazuje (referencira) na jedinku u drugoj relaciji. Dakle, strani ključ relacije je zapravo primarni ključ u relaciji na koju se referencira. Strani ključevi se koriste prilikom pretvaranja veza iz ER modela u relacije.

2.2.1. Detaljan opis relacijskog modela

Slijedi relacijski model dobiven iz prethodno opisanog ER modela baze podataka. Primarni ključevi relacija su podcrtani i podebljani. Pripadajući strani i alternativni ključevi navedeni su nakon svake relacije te je naznačeno na koju relaciju pokazuju. Alternativni ključ označen je s AK, a strani ključ oznakom FK.

- STUDENT – **id_student**, ime_student, prezime_student, mail_student, mobitel_student, oib_student, datum_rodenja_student, prebivaliste_student, fakultet_student, godina_studija, smjer_studija, datum_azuriranja, je_clan, je_aktivan_clan, korisnicko_ime, lozinka

$AK1 = \{korisnicko_ime\}, AK2 = \{oib_student\}$

- TIM – **id_tim**, naziv_tim, opis_tim, id_voditelj

$FK = \{id_voditelj\} \rightarrow STUDENT$

- PROJEKT – **id_projekt**, naziv_projekt, opis_projekt, pocetak_projekt, zavrsetak_projekt

- PARTNER – **id_partner**, naziv_partner, oib_partner, web_adresa_partner, mobitel_partner, mail_partner

$AK = \{oib_partner\}$

- SPONZORSKI PAKET – **id_sponzorski_paket**, naziv_sponzorski_paket, stavke, datum_stvaranja
- OKUPLJANJE – **id_okupljanje**, naziv_okupljanje, opis_okupljanje, pocetak_okupljanje, zavrsetak_okupljanje, je_formalno, id_tim
 $FK = \{id_tim\} \rightarrow TIM$
- MEDUNARODNI_DOGADAJ – **id_medunarodni_dogadaj**, naziv_medunarodni_dogadaj, opis_medunarodni_dogadaj, pocetak_medunarodni_dogadaj, zavrsetak_medunarodni_dogadaj, vrsta_medunarodni_dogadaj, kapacitet, cijena
- LOKALNI_OGRANAK – **id_lokalni_ogranak**, naziv_lokalni_ogranak, opis_lokalni_ogranak, id_drzava
 $FK = \{id_drzava\} \rightarrow DRZAVA$
- DRZAVA – **id_drzava**, naziv_drzava
- RACUN – **id_racun**, vrsta_racun, iznos_racun, vrijeme_racun, napomena, id_projekt, id_okupljanje
 $FK1 = \{id_projekt\} \rightarrow PROJEKT$
 $FK2 = \{id_okupljanje\} \rightarrow OKUPLJANJE$
- ULOGA – **id_uloga**, naziv_uloga
- prisustvuje – **id_student, id_okupljanje**
 $FK1 = \{id_student\} \rightarrow STUDENT$
 $FK2 = \{id_okupljanje\} \rightarrow OKUPLJANJE$
- prijava_na – **id_student, id_projekt**, je_prihvacen
 $FK1 = \{id_student\} \rightarrow STUDENT$
 $FK2 = \{id_projekt\} \rightarrow PROJEKT$
- je_organizator – **id_student, id_projekt**
 $FK1 = \{id_student\} \rightarrow STUDENT$
 $FK2 = \{id_projekt\} \rightarrow PROJEKT$

- putuje_na – **id student, id medunarodni dogadaj**, je_prihvacen, napomena

$FK1 = \{id_student\} \rightarrow STUDENT$

$FK2 = \{id_medunarodni_dogadaj\} \rightarrow MEDUNARODNI_DOGADAJ$

- je_clan – **id student, id tim**, napomena

$FK1 = \{id_student\} \rightarrow STUDENT$

$FK2 = \{id_tim\} \rightarrow TIM$

- sponzorira - **id projekt, id partner**, id_sponzorski_paket, iznos, napomena

$FK1 = \{id_projekt\} \rightarrow PROJEKT$

$FK2 = \{id_partner\} \rightarrow PARTNER$

$FK3 = \{id_sponzorski_paket\} \rightarrow SPONZORSKI_PAKET$

- organizira – **id medunarodni dogadaj, id lokalni ogranak**

$FK1 = \{id_medunarodni_dogadaj\} \rightarrow MEDUNARODNI_DOGADAJ$

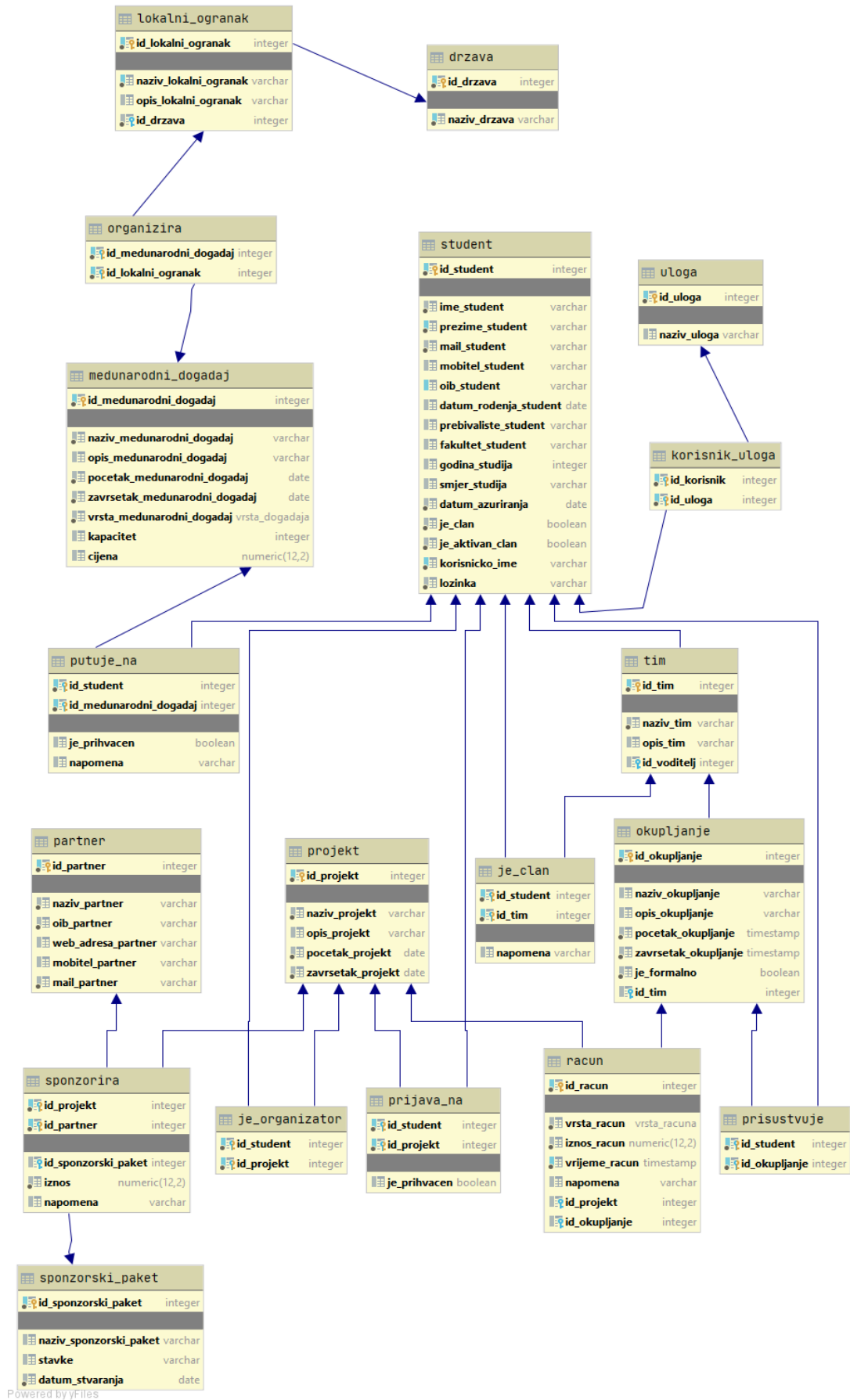
$FK2 = \{id_lokalni_ogranak\} \rightarrow LOKALNI_OGRANAK$

- korisnik_uloga – **id student, id uloga**

$FK1 = \{id_student\} \rightarrow STUDENT$

$FK2 = \{id_uloga\} \rightarrow ULOGA$

Na slici 2. prikazan je dijagram relacijskog modela baze podataka.



Slika 2. Dijagram relacijskog modela baze podataka

2.3. Integritet baze podataka

Baza podataka mora održavati integritet. To zapravo znači da podatci u njoj moraju u svakom trenutku biti konzistentni, u onom obliku u kojem je zamišljeno da budu spremljeni. Integritet se može narušiti na razne načine, no najčešći bi svakako bio pogrešan unos korisnika ili programskog servisa koji koristi bazu podataka. Sustav za upravljanje bazom podataka je zadužen da ne dođe do narušavanja integriteta. To se postiže postavljanjem raznih integritetskih ograničenja na attribute i same relacije.

Vrste integritetskih ograničenja su:

- Entitetski integritet
- Integritet ključa
- Domenski integritet
- Ograničenja *NULL* vrijednosti
- Referencijski integritet
- Opća integritetska ograničenja

Entitetski integritet osigurava da su u svakom trenutku jedinke nekog entiteta definirane. Praktično značenje je da primarni ključ nikada ne može poprimiti vrijednost *NULL*.

Integritet ključa osigurava da su sve jedinke nekog entiteta jednoznačno raspoznavljive. Praktično značenje je da u jednoj relaciji ne može postojati više n-torki (redaka) s jednakim ključem.

Domenski integritet osigurava da su vrijednosti atributa svake jedinke entiteta uvijek unutar definirane domene. U praksi se ovo postiže definiranjem tipa podataka za svaki atribut i po potrebi postavljanjem dodatnih ograničenja.

Ograničenja *NULL* vrijednosti definiraju attribute koji nikada ne smiju poprimiti nedefiniranu vrijednost (u PostgreSQL to je vrijednost *NULL*).

Referencijski integritet osigurava za svaki strani ključ koji ima definiranu vrijednost postoji jedinka na koju ključ pokazuje.

Na posljednjem mjestu su opća integritetska ograničenja. To su bilo koja ograničenja proizašla iz poslovne logike ili neke druge potrebe za njima. Ona se, u praksi, definiraju

naredbom CHECK unutar koje se definira neki logički izraz. Sustav za upravljanje bazom podataka odbija svaku izmjenu koja bi logički izraz učinila neistinitim.

Za implementaciju relacijskog modela iz poglavlja 2.2. potrebno je još definirati tip podatka za svaki atribut te neka ograničenja. Slijedi tablični prikaz s definiranim integritetskim ograničenjima za sve attribute svake relacije zadanog modela. Tipovi podataka su u skladu s PostgreSQL standardom.

Tablica 2. Pregled integritetskih ograničenja za relaciju STUDENT

ATRIBUT	TIP	OGRANIČENJA
id_student	SERIAL	Primarni ključ
ime_student	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
prezime_student	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
mail_student	VARCHAR	Mora biti ispravna email adresa Ne smije biti nedefiniran (NULL)
mobitel_student	VARCHAR	Mora sadržavati samo znakove za brojeve mobilnih telefona
oib_student	VARCHAR	Ključ Može sadržavati samo znamenke
datum_rođenja_student	DATE	Mora biti u prošlosti
prebivaliste_student	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
fakultet_student	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
godina_studija	INTEGER	Mora biti u intervalu [1, 5]
smjer_studija	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)

ATRIBUT	TIP	OGRANIČENJA
datum_azuriranja	DATE	Ne smije biti nedefiniran (NULL)
je_clan	BOOLEAN	Ne smije biti nedefiniran (NULL)
je_aktivan_clan	BOOLEAN	Ne smije biti nedefiniran (NULL)
korisnicko_ime	VARCHAR	Ne smije biti nedefiniran (NULL) ključ
lozinka	VARCHAR	Ne smije biti nedefiniran (NULL)

Tablica 3. Pregled integritetskih ograničenja za relaciju TIM

ATRIBUT	TIP	OGRANIČENJA
id_tim	SERIAL	Primarni ključ
naziv_tim	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
opis_tim	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
id_voditelj	INTEGER	Strani ključ na STUDENT

Tablica 4. Pregled integritetskih ograničenja za relaciju PROJEKT

ATRIBUT	TIP	OGRANIČENJA
id_projekt	SERIAL	Primarni ključ
naziv_projekt	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
opis_projekt	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
pocetak_projekt	DATE	Ne smije biti nedefiniran (NULL)
zavrsetak_projekt	DATE	Ne smije biti nedefiniran (NULL) mora biti nakon pocetak_projekt

Tablica 5. Pregled integritetskih ograničenja za relaciju PARTNER

ATRIBUT	TIP	OGRANIČENJA
id_partner	SERIAL	Primarni ključ
naziv_partner	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
oib_partner	VARCHAR	Mora sadržavati samo znamenke Ne smije biti nedefiniran (NULL) Ključ
web_adresa_partner	VARCHAR	Mora biti ispravna web adresa
mobitel_partner	VARCHAR	Mora sadržavati samo znakove za brojeve mobilnih telefona
mail_partner	VARCHAR	Ne smije biti nedefiniran (NULL) Mora biti ispravna email adresa

Tablica 6. Pregled integritetskih ograničenja za relaciju SPONZORSKI_PAKET

ATRIBUT	TIP	OGRANIČENJA
id_sponzorski_paket	SERIAL	Primarni ključ
naziv_sponzorski_paket	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
stavke	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
datum_stvaranja	DATE	Ne smije biti nedefiniran (NULL)

Tablica 7. Pregled integritetskih ograničenja za relaciju OKUPLJANJE

ATRIBUT	TIP	OGRANIČENJA
id_okupljanje	SERIAL	Primarni ključ
naziv_okupljanje	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
opis_okupljanje	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
pocetak_okupljanje	TIMESTAMP	Ne smije biti nedefiniran (NULL)
zavrsetak_okupljanje	TIMESTAMP	Ne smije biti nedefiniran (NULL) Mora biti nakon pocetak_okupljanje
je_formalno	BOOLEAN	Ne smije biti nedefiniran (NULL)
id_tim	INTEGER	Strani ključ

Tablica 8. Pregled integritetskih ograničenja za relaciju MEDUNARODNI_DOGADAJ

ATRIBUT	TIP	OGRANIČENJA
id_okupljanje	SERIAL	Primarni ključ
naziv_medunarodni_dogadaj	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
opis_medunarodni_dogadaj	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
pocetak_medunarodni_dogadaj	DATE	Ne smije biti nedefiniran (NULL)
zavrsetak_medunarodni_dogadaj	DATE	Ne smije biti nedefiniran (NULL) Mora biti nakon pocetak_medunarodni_dogadaj
vrsta_medunarodni_dogadaj	vrsta_dogadaja	Ne smije biti nedefiniran (NULL)
kapacitet	INTEGER	Mora biti > 0
cijena	NUMERIC(12, 2)	Mora biti >= 0

Napomena: vrsta_dogadaja kao tip podatka je enumeracija koja ima jednu od sljedećih vrijednosti: 'RADIONICA', 'RAZMJENA', 'NAPREDNA_RADIONICA', 'OPERACIJSKI_DOGADAJ', 'MOTIVACIJSKI_VIKEND'.

Tablica 9. Pregled integritetskih ograničenja za relaciju DRZAVA

ATRIBUT	TIP	OGRANIČENJA
id_drzava	SERIAL	Primarni ključ
naziv_drzava	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)

Tablica 10. Pregled integritetskih ograničenja za relaciju LOKALNI_OGRANAK

ATRIBUT	TIP	OGRANIČENJA
id_lokalni_ogranak	SERIAL	Primarni ključ
naziv_lokalni_ogranak	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...) Ne smije biti nedefiniran (NULL)
opis_lokalni_ogranak	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
id_drzava	INTEGER	Strani ključ

Tablica 11. Pregled integritetskih ograničenja za relaciju RACUN

ATRIBUT	TIP	OGRANIČENJA
id_racun	SERIAL	Primarni ključ
vrsta_racun	vrsta_racuna	Ne smije biti nedefiniran (NULL)
iznos_racun	NUMERIC(12, 2)	Ne smije biti nedefiniran (NULL) Mora biti > 0
vrijeme_racun	TIMESTAMP	Ne smije biti nedefiniran (NULL)
napomena	VARCHAR	Ne smije biti „prazan” (samo razmak, tabulator...)
id_projekt	INTEGER	Strani ključ
id_okupljanje	INTEGER	Strani ključ

Napomena: vrsta_racuna kao tip podatka je enumeracija koja ima jednu od sljedećih vrijednosti: 'PRIHOD', 'RASHOD'.

Tablica 12. Pregled integritetskih ograničenja za relaciju ULOGA

ATRIBUT	TIP	OGRANIČENJA
id_uloga	SERIAL	Primarni ključ
naziv_uloga	VARCHAR	

Tablica 13. Pregled integritetskih ograničenja za relaciju korisnik_uloga

ATRIBUT	TIP	OGRANIČENJA
id_korisnik	INTEGER	Primarni ključ Strani ključ
id_uloga	INTEGER	Primarni ključ Strani ključ

Tablica 14. Pregled integritetskih ograničenja za relaciju prisustvuje

ATRIBUT	TIP	OGRANIČENJA
id_student	INTEGER	Primarni ključ Strani ključ
id_okupljanje	INTEGER	Primarni ključ Strani ključ

Tablica 15. Pregled integritetskih ograničenja za relaciju prijava_na

ATRIBUT	TIP	OGRANIČENJA
id_student	INTEGER	Primarni ključ Strani ključ
id_projekt	INTEGER	Primarni ključ Strani ključ
je_prihvacen	BOOLEAN	

Tablica 16. Pregled integritetskih ograničenja za relaciju je_organizator

ATRIBUT	TIP	OGRANIČENJA
id_student	INTEGER	Primarni ključ Strani ključ
id_projekt	INTEGER	Primarni ključ Strani ključ

Tablica 17. Pregled integritetskih ograničenja za relaciju putuje_na

ATRIBUT	TIP	OGRANIČENJA
id_student	INTEGER	Primarni ključ Strani ključ
id_medunarodni_dogadaj	INTEGER	Primarni ključ Strani ključ
je_prihvacen	BOOLEAN	
napomena	VARCHAR	

Tablica 18. Pregled integritetskih ograničenja za relaciju je_clan

ATRIBUT	TIP	OGRANIČENJA
id_student	INTEGER	Primarni ključ Strani ključ
id_tim	INTEGER	Primarni ključ Strani ključ
napomena	VARCHAR	

Tablica 19. Pregled integritetskih ograničenja za relaciju sponzorira

ATRIBUT	TIP	OGRANIČENJA
id_projekt	INTEGER	Primarni ključ Strani ključ
id_partner	INTEGER	Primarni ključ Strani ključ
id_sponzorski_paket	INTEGER	Strani ključ
iznos	NUMERIC(12, 2)	Ne smije biti nedefiniran (NULL) Mora biti ≥ 0
napomena	VARCHAR	

Tablica 20. Pregled integritetskih ograničenja za relaciju organizira

ATRIBUT	TIP	OGRANIČENJA
id_medunarodni_dogadaj	INTEGER	Primarni ključ Strani ključ
id_lokalni_ogranak	INTEGER	Primarni ključ Strani ključ

Integritet podataka se ponekad ne može postići prethodno navedenim ograničenjima. U nekim slučajevima je potrebno koristiti okidače i procedure (*eng. triggers and procedures*). Okidači su događaji koje sustav za upravljanje bazom podataka „osluškuje“ te kada se dogode, sustav pokreće unaprijed definiranu proceduru (funkciju).

U ovoj bazi podataka bit će implementiran okidač prilikom ažuriranja tablice STUDENT. Sustav će prilikom ažuriranja atribut datum_azuriranja postaviti na trenutni datum sustava bez obzira što je poslano u naredbi za ažuriranje.

2.4. Implementacija baze podataka

Nakon što je u prethodnim poglavljima definirana detaljna shema baze podataka i ograničenja, preostalo je implementirati sve navedeno. Relacije i ograničenja opisat će se standardnim jezikom baza podataka SQL. Procedure i okidači implementirani su u proceduralnom jeziku PL/pgSQL (Procedural Language/PostgreSQL).

Za implementaciju baze podataka korišten je lokalno instaliran PostgreSQL sustav za upravljanje bazom podataka, pgAdmin 4 platforma za administraciju i razvoj PostgreSQL baza podataka te JetBrains DataGrip razvojno okruženje za baze podataka.

2.4.1. Implementacija relacija

Sve navedene relacije su opisane jezikom SQL i kreirane od strane sustava za upravljanje bazom podataka. Slijede isječki koda implementacije relacija.

```
create table student
(
    id_student      SERIAL      constraint student_pk primary key,
    ime_student     VARCHAR CHECK (trim(ime_student) NOT LIKE '')
                                not null,
    prezime_student VARCHAR CHECK (trim(prezime_student) NOT LIKE '')
                                not null,
    mail_student    VARCHAR CHECK (mail_student LIKE '_%@%.%.%' ) not null,
    mobitel_student VARCHAR CHECK ( mobitel_student SIMILAR TO
                                '([0-9]|\\+|\\(|\\)|\\s)+'),
    oib_student     VARCHAR UNIQUE CHECK ( oib_student SIMILAR TO
                                '[0-9]+' ),
    datum_rodenja_student DATE
                                CHECK ( datum_rodenja_student < current_date ),
    prebivaliste_student VARCHAR
                                CHECK ( trim(prebivaliste_student) NOT LIKE ''),
    fakultet_student VARCHAR CHECK ( trim(fakultet_student) NOT LIKE ''),
    godina_studija  INTEGER
                                CHECK ( godina_studija < 6 AND godina_studija > 0 ),
    smjer_studija   VARCHAR CHECK ( trim(smjer_studija) NOT LIKE ''),
    datum_azuriranja DATE      default current_date not null,
    je_clan         BOOLEAN default true           not null,
```

```

je_aktivan_clan      BOOLEAN default false      not null,

korisnicko_ime       VARCHAR UNIQUE              not null,

lozinka              VARCHAR default
'$2y$12$Tw3uvmXE6aEDUo8w2PXBve.eeVLZlU1AGMsWSso00L2gZ0XO9CkaK' not null
);

```

Kod 1. Implementacija relacije STUDENT

```

create table tim
(
    id_tim            SERIAL
        constraint tim_pk
            primary key,

    naziv_tim        VARCHAR not null CHECK ( trim(naziv_tim) NOT LIKE '' ),

    opis_tim         VARCHAR CHECK ( trim(opis_tim) NOT LIKE '' ),

    id_voditelj int
        constraint tim_voditelj_fk
            references student
            on delete set null
);

```

Kod 2. Implementacija relacije TIM

```

create table projekt
(
    id_projekt        SERIAL
        constraint projekt_pk
            primary key,

    naziv_projekt     VARCHAR not null
        CHECK ( trim(naziv_projekt) NOT LIKE '' ),

    opis_projekt      VARCHAR CHECK ( trim(opis_projekt) NOT LIKE '' ),

    pocetak_projekt   DATE      not null,

    zavrsetak_projekt DATE      not null
        CHECK ( zavrsetak_projekt > pocetak_projekt )
);

```

Kod 3. Implementacija relacije PROJEKT


```

create table partner
(
    id_partner          SERIAL
        constraint partner_pk
        primary key,

    naziv_partner       VARCHAR not null
        CHECK ( trim(naziv_partner) NOT LIKE '' ),

    oib_partner         VARCHAR UNIQUE not null
        CHECK ( oib_partner SIMILAR TO '[0-9]+' ),

    web_adresa_partner  VARCHAR
        CHECK ( web_adresa_partner SIMILAR TO '^[^s(["<, >]*\.[^s[">, <]*' ),

    mobitel_partner     VARCHAR
        CHECK ( mobitel_partner SIMILAR TO '([0-9]|\+|\(|\)|\s)+' ),

    mail_partner        VARCHAR not null
        CHECK ( mail_partner LIKE '%@%.%' )
);

```

Kod 4. Implementacija relacije PARTNER

```

create table sponzorski_paket
(
    id_sponzorski_paket SERIAL
        constraint sponzorski_paket_pk
        primary key,

    naziv_sponzorski_paket VARCHAR
        CHECK ( trim(naziv_sponzorski_paket) NOT LIKE '' ),

    stavke               VARCHAR CHECK ( trim(stavke) NOT LIKE '' ),

    datum_stvaranja      DATE default current_date not null
);

```

Kod 5. Implementacija relacije SPONZORSKI_PAKET

```

create table okupljanje
(
    id_okupljanje       SERIAL
        constraint okupljanje_pk
        primary key,

    naziv_okupljanje    VARCHAR CHECK ( trim(naziv_okupljanje) NOT LIKE '' ),

    opis_okupljanje     VARCHAR CHECK ( trim(opis_okupljanje) NOT LIKE '' ),

    pocetak_okupljanje  TIMESTAMP not null,

    zavrsetak_okupljanje TIMESTAMP not null
        CHECK ( zavrsetak_okupljanje > pocetak_okupljanje ),

    je_formalno         BOOLEAN not null,

```

```

        id_tim          int
        constraint okupljanje_tim_fk
        references tim
        on delete cascade
    );

```

Kod 6. Implementacija relacije OKUPLJANJE

```

create table medunarodni_dogadaj
(
    id_medunarodni_dogadaj SERIAL constraint medunarodni_dogadaj_pk
        primary key,

    naziv_medunarodni_dogadaj VARCHAR not null
        CHECK ( trim(naziv_medunarodni_dogadaj) NOT LIKE '' ),

    opis_medunarodni_dogadaj VARCHAR
        CHECK ( trim(opis_medunarodni_dogadaj) NOT LIKE '' ),

    pocetak_medunarodni_dogadaj DATE not null,

    zavrsetak_medunarodni_dogadaj DATE not null
    CHECK ( zavrsetak_medunarodni_dogadaj > pocetak_medunarodni_dogadaj ),

    vrsta_medunarodni_dogadaj vrsta_dogadaja not null,

    kapacitet INT CHECK ( kapacitet > 0 ),

    cijena NUMERIC(12, 2) CHECK ( cijena >= 0 )
);

```

Kod 7. Implementacija relacije MEDUNARODNI_DOGADAJ

```

create table drzava
(
    id_drzava SERIAL
        constraint drzava_pk
        primary key,

    naziv_drzava VARCHAR not null UNIQUE
        CHECK ( trim(naziv_drzava) NOT LIKE '' )
);

```

Kod 8. Implementacija relacije DRZAVA

```

create table lokalni_ogranak
(
    id_lokalni_ogranak    SERIAL
        constraint lokalni_ogranak_pk
            primary key,

    naziv_lokalni_ogranak VARCHAR not null UNIQUE
        CHECK ( trim(naziv_lokalni_ogranak) NOT LIKE '' ),

    opis_lokalni_ogranak  VARCHAR
        CHECK ( trim(opis_lokalni_ogranak) NOT LIKE '' ),

    id_drzava             INT         not null
        constraint lokalni_ogranak_drzava_fk
            references drzava
            on delete restrict
);

```

Kod 9. Implementacija relacije LOKALNI_OGRANAK

```

create table racun
(
    id_racun              SERIAL
        constraint racun_pk
            primary key,

    vrsta_racun           vrsta_racuna          not null,

    iznos_racun           NUMERIC(12, 2)        not null CHECK ( iznos_racun > 0 ),

    vrijeme_racun         TIMESTAMP default current_timestamp not null,

    napomena              VARCHAR CHECK ( trim(napomena) NOT LIKE '' ),

    id_projekt            INT                    default null
        constraint racun_projekt_fk
            references projekt
            on delete set null,

    id_okupljanje         INT                    default null
        constraint racun_okupljanje_fk
            references okupljanje
            on delete set null
);

```

Kod 10. Implementacija relacije RACUN

```

create table uloga
(
    id_uloga             SERIAL
        constraint uloga_pk
            primary key,

    naziv_uloga          VARCHAR default NULL
);

```

Kod 11. Implementacija relacije ULOGA

```

create table korisnik_uloga
(
    id_korisnik INTEGER not null
        constraint korisnik_uloga_to_student_fk
        references student
        on delete cascade,

    id_uloga    INTEGER not null
        constraint korisnik_uloga_to_uloga_fk
        references uloga (id_uloga)
        on delete cascade,

    constraint korisnik_uloga_pk
        primary key (id_korisnik, id_uloga)
);

```

Kod 12. Implementacija relacije korisnik_uloga

```

create table prisustvuje
(
    id_student    INT
        constraint student_prisustvuje_fk
        references student
        on delete cascade,

    id_okupljanje INT
        constraint okupljanje_prisustvuje_fk
        references okupljanje
        on delete cascade,

    constraint prisustvuje_pk
        primary key (id_student, id_okupljanje)
);

```

Kod 13. Implementacija relacije prisustvuje

```

create table prijava_na
(
    id_student    INT
        constraint student_prijavana_fk references student on delete cascade,

    id_projekt    INT
        constraint projekt_prijavana_fk references project on delete cascade,

    je_prihvacen BOOLEAN,
    constraint prijavana_pk primary key (id_student, id_projekt)
);

```

Kod 14. Implementacija relacije prijava_na

```

create table je_organizator
(
    id_student INT
        constraint student_jeorganizator_fk
        references student
        on delete cascade,

    id_projekt INT
        constraint projekt_jeorganizator_fk
        references projekt
        on delete cascade,

    constraint jeorganizator_pk
        primary key (id_student, id_projekt)
);

```

Kod 15. Implementacija relacije je_organizator

```

create table putuje_na
(
    id_student INT
        constraint student_putujena_fk
        references student
        on delete cascade,

    id_medunarodni_dogadaj INT
        constraint medunarodniDogadaj_putujena_fk
        references medunarodni_dogadaj
        on delete cascade,

    je_prihvacen BOOLEAN,

    napomena VARCHAR,

    constraint putujena_pk
        primary key (id_student, id_medunarodni_dogadaj)
);

```

Kod 16. Implementacija relacije putuje_na

```

create table je_clan
(
    id_student INT
        constraint student_jeclan_fk
        references student
        on delete cascade,

    id_tim INT
        constraint tim_jeclan_fk
        references tim
        on delete cascade,

    napomena VARCHAR,

    constraint jeclan_pk
        primary key (id_student, id_tim)
);

```

Kod 17. Implementacija relacije je_clan

```

create table sponzorira
(
    id_projekt          INT
        constraint projekt_sponzorira_fk
        references projekt
        on delete cascade,

    id_partner          INT
        constraint partner_sponzorira_fk
        references partner
        on delete cascade,

    id_sponzorski_paket INT
        constraint sponzorskiPaket_sponzorira_fk
        references sponzorski_paket
        on delete set null,

    iznos               NUMERIC(12, 2) not null CHECK ( iznos >= 0 ),

    napomena            VARCHAR,

    constraint sponzorira_pk
        primary key (id_projekt, id_partner)
);

```

Kod 18. Implementacija relacije sponzorira

```

create table organizira
(
    id_medunarodni_dogadaj INT
        constraint medunarodniDogadaj_organizira_fk
        references medunarodni_dogadaj
        on delete cascade,

    id_lokalni_ogranak     INT
        constraint lokalniOgranak_organizira_fk
        references lokalni_ogranak
        on delete cascade,

    constraint organizira_pk
        primary key (id_medunarodni_dogadaj, id_lokalni_ogranak)
);

```

Kod 19. Implementacija relacije organizira

2.4.2. Implementacija okidača i procedura

U prijašnjem poglavlju je spomenut okidač i procedura za očuvanje integriteta atributa *datum_azuriranja* u relaciji STUDENT. Zbog praktičnosti i automatizacije praćenja računa, implementiran je i okidač na unos novog sponzorstva koji pokreće proceduru koja kreira novi račun za to sponzorstvo. Slijedi isječak koda implementacije ovih okidača i procedura.

```

create function azuriranje_studenta() returns trigger
as
$$
begin
    NEW.datum_azuriranja = current_date;
    RETURN NEW;

end;
$$ LANGUAGE plpgsql;

create trigger update_student
before update
on student
for each ROW
execute procedure azuriranje_studenta();

```

Kod 20. Implementacija okidača i procedure očuvanja integriteta atributa datum_azuriranja

```

create function racun_iz_sponzorstva() returns trigger
as
$$
DECLARE
    partner_name VARCHAR;
    projekt_name VARCHAR;
begin
    IF NEW.iznos = 0
    THEN
        RETURN NEW;
    ELSE
        SELECT partner.naziv_partner INTO partner_name
            FROM partner WHERE partner.id_partner = NEW.id_partner;

        SELECT projekt.naziv_projekt INTO projekt_name
            FROM projekt WHERE projekt.id_projekt = NEW.id_projekt;

        INSERT INTO racun(vrsta_racun, iznos_racun, napomena, id_projekt)
            VALUES ('PRIHOD', NEW.iznos,
                'Račun od sponzorstva; SPONZOR: ' || partner_name || ';
                PROJEKT: ' || projekt_name, NEW.id_projekt);

        RETURN NEW;
    END IF;
end ;
$$ LANGUAGE plpgsql;

create trigger insert_sponzorira
after insert
on sponzorira
for each ROW
execute procedure racun_iz_sponzorstva();

```

Kod 21. Implementacija okidača i procedure za kreiranje računa nakon unosa novog sponzorstva

2.5. Podatci za testiranje i daljnji razvoj

Za potrebe testiranja i razvoja web aplikacije, u bazu podataka je potrebno pohraniti testne podatke. Pomoću generatora podataka na web stranicama *generatedata.com* i *mockaroo.com* napravljena je skripta SQL naredbi koje u bazu dodaju testne podatke. Isječak koda primjera jedne od INSERT naredbi slijedi u nastavku.

```
INSERT INTO "partner" (naziv_partner, oib_partner, web_adresa_partner,
mobitel_partner, mail_partner)
VALUES ('Photojam', 19501365,
'sogou.com/mauris/vulputate/elementum/nullam.aspx', '+502 131 406 0513',
'rbedenham0@weibo.com')
, ('Ailane', 34458992,
'bloglovin.com/erat/vestibulum/sed/magna/at/nunc/commodo.xml', '+598 898
990 7238',
'wgodbold1@exblog.jp')
, ('Aimbu', 54600823, 'smh.com.au/molestie/sed.jpg', '+33 114 477
8458', 'mrandal2@cam.ac.uk')
, ('Abatz', 44286904, 'amazon.co.jp/diam.xml', '+49 777 243 5781',
'tcudde3@a8.net')
, ('Zoomcast', 76147533,
'reference.com/at/nulla/suspendisse/potenti/cras.html', '+62 485 325
3220',
'klearmond4@go.com')
, ('Thoughtbridge', 38286680, 'bluehost.com/nec/molestie.png', '+30
454 205 4294', 'bnovichenko5@amazonaws.com')
, ('Oyundu', 24463120,
'blogs.com/iaculis/congue/vivamus/metus/arcu.xml', '+62 613 830 2471',
'rmarkos6@blogspot.com')
...
...
...
```

Kod 22. Isječak koda za pohranjivanje testnih podataka u bazu

3. Web aplikacija

Potrebno je razviti programsku potporu koja će koristiti bazu podataka opisanu u poglavlju 2. u skladu sa specifikacijom zahtjeva iz poglavlja 1. Web aplikacija je praktično rješenje zbog jednostavnosti pristupa i mogućnosti korištenja bez instalacije dodatnih programa od strane korisnika (osim web preglednika). Također, web aplikaciju je moguće prilagoditi da bude upotrebljiva i na prijenosnim uređajima poput mobilnih telefona, no zbog opsežnosti taj dio neće biti razmatran u ovome radu.

3.1. Arhitektura web aplikacije

U implementaciji web aplikacije korištena je MVC (*eng. Model-View-Controller*) arhitektura. Arhitekturom model-pogled-upravljač želi se postići raspodjela „odgovornosti“ za pojedine radnje u aplikaciji. Neke prednosti MVC arhitekture su:

- Lako se nadograđuje i održava
- Pogodna je za rad u procesorskim sustavima s više jezgri
- Odvojenost poslovne logike, korisničkog sučelja i načina pristupa podacima.

Aplikacija je podijeljena na tri glavne komponente čiji nazivi i čine ime arhitekture. To su model, pogled i upravljač.

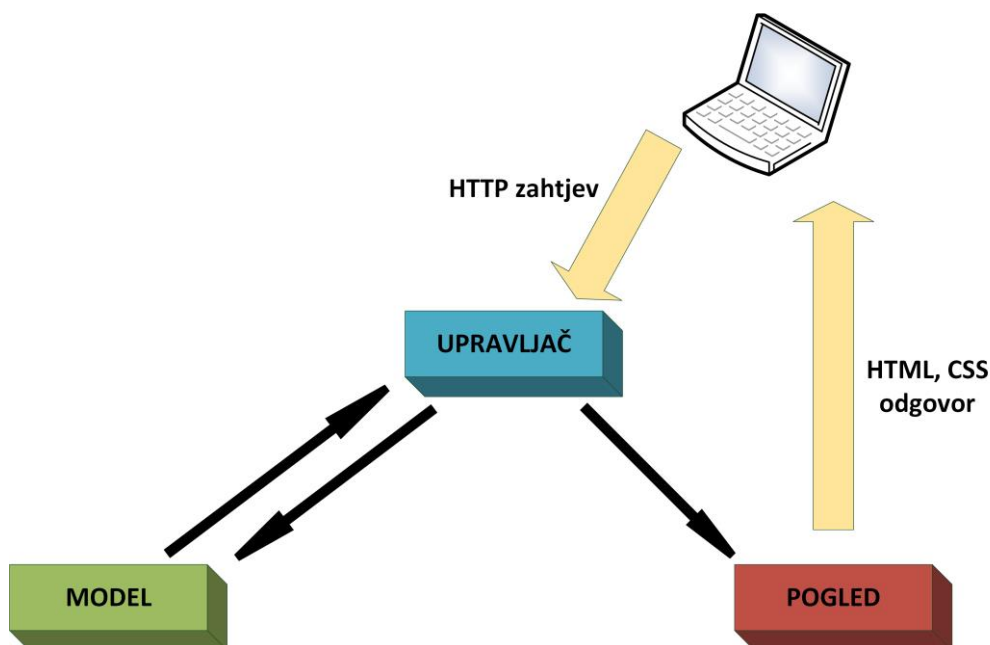
Model čini sav kod koji služi za baratanje podacima. Tu se nalaze naredbe za komunikacijom s bazom podataka, ali i razne strukture podataka aplikacije koje služe za prijenos podataka između komponenti ili izvan aplikacije.

Pogled sadrži sav kod koji služi za komunikaciju aplikacije s korisnikom. U slučaju web aplikacija to su najčešće web stranice, ali mogu biti i razne metode za izvoz podataka (npr. u PDF format). Ovo je ujedno jedina komponenta ove arhitekture vidljiva korisniku.

Upravljač čini kod koji upravlja tokom aplikacije. U slučaju web aplikacije, upravljač prima i šalje HTML zahtjeve. U pravilu upravljač poziva model kako bi primio ili poslao podatke, a zatim te podatke prezentira korisniku koristeći određeni pogled.

Web aplikacija je implementirana u programskom jeziku Java koristeći Spring framework (uključujući i Spring MVC).

Na slici 3. prikazan je shematski prikaz MVC arhitekture web aplikacije i njene interakcije s korisničkim računalom.



Slika 3. Shematski prikaz MVC arhitekture

3.1.1. Implementacija upravljača

U praksi se implementira upravljač za svaki entitet tj. relaciju koja se želi izmjenjivati. Pošto je aplikacija implementirana korištenjem Spring MVC framework, anotacijom je označena klasa upravljač. Također, pomoću anotacija su definirane veze na koje je potrebno slati HTTP zahtjeve, kao i vrsta zahtjeva (POST ili GET). Objektu klase `Model` može se pristupiti iz pogleda, te se on koristi za prijenos podataka između klijentske strane i serverske strane aplikacije (*eng. backend and frontend*). Osim pomoću objekta klase `Model`, podatci se mogu izmjenjivati i postavljanjem parametara HTTP zahtjeva. Servisi predstavljaju sloj poslovne logike i dio su modela ako razmatramo MVC arhitekturu. Slijedi isječak koda iz klase upravljača.

```

@Controller
@RequestMapping("/student")
public class StudentController {

    @Autowired
    private StudentService studentService;
    @Autowired
    private RoleService roleService;
    @Autowired
    private GatheringService gatheringService;
    ...
    @GetMapping("/list")
    public String listStudents(Model model){

        List<Student> students = studentService.getStudents();
        model.addAttribute("studentService", studentService);
        model.addAttribute("students", students);
        model.addAttribute("mappingPath", "");
        model.addAttribute("studentButton", "");
        model.addAttribute("memberButton", "display: none");

        return "list-students";
    }

    @GetMapping("/showFormForAdd")
    public String showFormForAdd(Model model){

        Student student = new Student();
        model.addAttribute("student", student);
        model.addAttribute("disabled_edit", false);
        model.addAttribute("saveButton", "visible");
        model.addAttribute("editButton", "hidden");
        model.addAttribute("showGatherings", "display: none");

        return "student-form";
    }

    @PostMapping("/save")
    public String saveStudent(@Valid @ModelAttribute("student") Student
student, BindingResult bindingResult, Model model){

        if (bindingResult.hasErrors()){

            model.addAttribute("disabled_edit", false);
            model.addAttribute("saveButton", "visible");
            model.addAttribute("editButton", "hidden");
            model.addAttribute("showGatherings", "display: none");

            return "student-form";
        }
        if(studentService.getStudent(student.getId()) == null)
            studentService.createStudent(student);
        else
            studentService.editStudent(student);

        return "redirect:/student/list";
    }

    ...
}

```

Kod 23. Isječak koda implementacije klase StudentController

3.1.2. Implementacija modela

Model se također, u pravilu, implementira za svaku relaciju s čijim podacima je potrebno upravljati. U web aplikaciji model se sastoji od dva dijela:

- Sloj poslovne logike
- Sloj pristupa podacima

Ostali slojevi aplikacije koriste isključivo metode poslovne logike, a onda sloj poslovne logike „komunicira“ sa slojem za pristup podacima. Na taj način moguće je lako promijeniti izvor podataka ili čak imati podatke iz više izvora podataka (npr. više baza podataka).

Klase koje predstavljaju sloj poslovne logike nazvane su servisima. U ovom sloju se obavljaju sve potrebne operacije nad podacima koje proizlaze iz neke logike koja u pravilu nije povezana sa samom implementacijom, već s problemom koji se rješava. U ovom sloju se obavljaju i akcije prilagodbe formata podataka za idući sloj, ili preslikavanje iz objekata jedne klase u objekte druge klase zbog potrebe za korištenjem klasa za prijenos podataka (eng. *Data Transfer Object*). Slijedi isječak koda iz klase servisa.

```
@Service
public class StudentServiceImpl implements StudentService{

    @Autowired
    private StudentDAO studentDAO;

    @Autowired
    private ModelMapper modelMapper;

    @Autowired
    private RoleService roleService;

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    @Override
    @Transactional
    public List<Student> getStudents() {
        return studentDAO.getStudents();
    }

    @Override
    @Transactional
```

```

    public void createStudent(Student student) {

        student.setUsername(student.getName() +
ThreadLocalRandom.current().nextInt(1, 9999 + 1));
        student.setPassword(passwordEncoder.encode("1234"));
        List<Role> roles = student.getRoles();
        roles.add(roleService.getByName("ROLE_USER"));
        student.setRoles(roles);

        studentDAO.saveStudent(student);
    }

    @Override
    @Transactional
    public void editStudent(Student student) {

        Student original = getStudent(student.getId());
        modelMapper.map(student, original);

        studentDAO.saveStudent(original);
    }

    @Override
    @Transactional
    public Student getStudent(int id) {
        return studentDAO.getStudent(id);
    }

```

Kod 24. Isječak koda implementacije klase StudentService

Klase koje predstavljaju sloj pristupa podacima nazvane su kraticom od engleskog izraza *Data Access Object*. U ovom sloju obavlja se komunikacija s bazom podataka. Posredstvom Hibernate framework metoda, izvršavaju se upiti nad bazom te se rezultati preslikavaju u odgovarajuće objekte. Slijedi isječak koda iz klase sloja pristupa podacima.

```

@Repository
public class StudentDAOImpl implements StudentDAO{

    @Autowired
    private SessionFactory sessionFactory;

    @Override
    public List<Student> getStudents() {

        Session session = sessionFactory.getCurrentSession();
        Query<Student> studentQuery = session.createQuery("from Student
                                                         order by surname", Student.class);

        return studentQuery.getResultList();
    }

    @Override
    public void saveStudent(Student student) {

        Session session = sessionFactory.getCurrentSession();

        session.saveOrUpdate(student);
    }

    @Override
    public Student getStudent(int id) {

        Session session = sessionFactory.getCurrentSession();

        return session.get(Student.class, id);
    }

    @Override
    public Student findByUsername(String username) {

        Session session = sessionFactory.getCurrentSession();
        Query<Student> studentQuery = session.createQuery("from Student
                                                         where username =:uName", Student.class);
        studentQuery.setParameter("uName", username.trim());

        Student student = null;
        try {
            student = studentQuery.getSingleResult();
        } catch (Exception e) {
            student = null;
        }

        return student;
    }
}

```

Kod 25. Isječak koda implementacije klase StudentDAO

3.1.3. Implementacija pogleda

Implementaciju pogleda čine JavaServer Pages datoteke. One se sastoje uglavnom isključivo od koda u HTML i CSS jezicima. Ipak, moguće je pokretati i neke naredbe serverske strane aplikacije čime se donekle narušava MVC arhitektura. Za prijenos podataka između upravljača i pogleda koriste se parametri HTML zahtjeva ili već spomenuta klasa `Model`. Slijedi isječak koda datoteke jednog pogleda.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>

<html>
<head>
  <title>Popis studenata</title>

  <link
href="${pageContext.request.contextPath}/resources/css/style.css"
rel="stylesheet" type="text/css">
</head>
<body>

<div class="navigation">

  <div class="my-menu">
    <!-- Logout button and Homepage button-->
    <form:form action="${pageContext.request.contextPath}/logout"
method="POST">

      <input type="submit" value="Odjavi se" class="logout-button
my-menu-button"/>
    </form:form>

    <input type="button" value="Početna stranica"
onclick="window.location='/'; return false;"
class="logout-button my-menu-button">
  </div>
</div>

<div id="wrapper">
  <div id="header">
    <h2>Aplikacija za studentsku udrugu</h2>
  </div>
</div>

<div id="container">
  <div id="content">

    <security:authorize access="hasAnyRole('BOARD_MEMBER')">
      <input type="button" value="Dodaj studenta"
onclick="window.location.href='showFormForAdd'; return false;"
class="add-button" style="${studentButton}">

      <input type="button" value="Dodaj člana"
onclick="window.location.href='formAddMember?id=${param.get('id')}'";
```

```

return false;"
        class="add-button" style="${memberButton}" ">
    </security:authorize>

    <table>
    <tr>
        <th>Ime i prezime</th>
        <th>Fakultet</th>
        <th>Godina studija</th>
        <th>Broj okupljanja</th>
        <th>Broj putovanja</th>
        <th></th>
    </tr>

    <c:forEach var="tempStudent" items="${students}">

        <c:url var="detailsLink" value="/student/details">
            <c:param name="studentId"
value="${tempStudent.id}"/>
        </c:url>

        <c:url var="deleteMemberLink"
value="/${mappingPath}/removeMember">
            <c:param name="id" value="${param.get('id')}" />
            <c:param name="MemberId"
value="${tempStudent.id}"/>
        </c:url>

        <tr>
            <td>${tempStudent.fullName}</td>
            <td>${tempStudent.faculty}</td>
            <td>${tempStudent.yearOfStudy}</td>

            <td>${studentService.getNumberOfGatheringsParticipated(tempStudent.id)}</td>

            <td>${studentService.getNumberOfEventsParticipated(tempStudent.id)}</td>
            <td>
                <security:authorize
access="hasAnyRole('BOARD_MEMBER') ">
                    <a href="${detailsLink}">Otvori profil</a> |
                <a href="${deleteMemberLink}" style="${memberButton}">Ukloni</a>
                </security:authorize>
            </td>
        </tr>

    </c:forEach>
    </table>

</div>
</div>
</body>
</html>

```

Kod 26. Isječak koda implementacije pogleda list-students

3.2. Korištene tehnologije

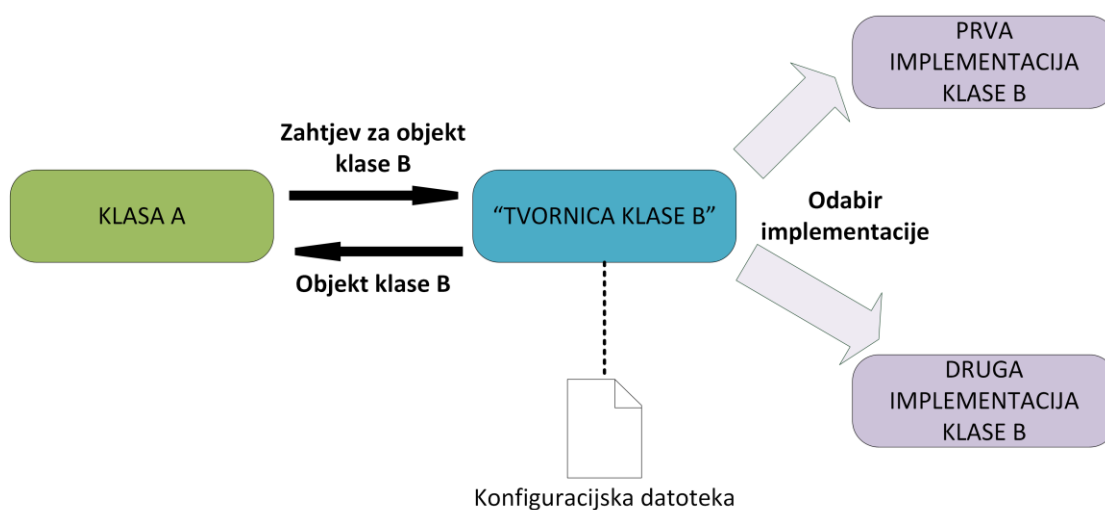
Web aplikacija implementirana je u objektno orijentiranoj paradigmi koristeći programski jezik Java. Prikaz korisničkog sučelja ostvaren je pomoću tehnologija HTML, CSS i JavaServer Pages. Korištene su i sljedeće tehnologije:

- Spring framework
- Hibernate ORM framework
- Apache Maven

Spring framework dolazi sa mnogo biblioteka za razne namjene. Cilj mu je pojednostaviti i skratiti vrijeme razvoja aplikacije. Najvažnije značajke su mu:

- Inverzija upravljanja tokom programa (*eng. Inversion of control*)
- „Ubacivanje“ potrebnih objekata (*eng. Dependency injection*)

Inverzija upravljanja tokom programa ima za cilj što više smanjiti ovisnost između klasa u programu. Zove se tako jer je postupak kojim se stvaraju objekti neke klase inverzan tradicionalnom proceduralnom programiranju. U suštini, to je postupak kojim stvaranje novih objekata neke klase ne ostvarujemo pozivanjem konstruktora te klase. Umjesto toga kreira se dodatna klasa koja služi kao „tvornica objekata“. Ona poziva konstruktor, po potrebi izvršava dodatne radnje prije i poslije stvaranja objekta te vraća objekt spreman za korištenje. Velika prednost je i što može postojati više implementacija tražene klase, a „tvornica objekata“ na temelju konfiguracijskih datoteka vraća primjerenu implementaciju. Slika 4. prikazuje shematski prikaz inverzije upravljanja tokom programa.



Slika 4. Shematski prikaz inverzije upravljanja tokom programa

Inverzija upravljanja tokom programa je upravo i potrebna za ostvarivanje sljedeće značajke. Ukoliko postoji neka klasa čiji objekti sadrže (ili ovise) o objektima neke druge klase, programer, koristeći Spring framework, ne mora eksplicitno kreirati te objekte druge klase. „Tvornica objekata“ će automatski od drugih „tvornica objekata“ zatražiti objekte koji su potrebni te će ih „ubaciti“ i vratiti traženi objekt spreman za korištenje. To je „ubacivanje“ potrebnih objekata (*eng. Dependency injection*).

Hibernate ORM framework služi za preslikavanje relacija u objekte i obrnuto (*eng. Object/Relational Mapping*). Pomoću njega modelirane su Java klase koje predstavljaju entitete u opisanom modelu. Posebnim oznakama naznačene su veze atributa relacije s atributima klase i pripadajuća ograničenja. Ova tehnologija omogućava da se prividno iz baze dohvaćaju objekti koji predstavljaju n-torku u bazi podataka te se rad s podacima svodi na rad s Java objektima. Slijedi isječak koda implementacije jedne Java klase entiteta koristeći tehnologiju Hibernate ORM framework, a na slici 5. je prikazan dijagram razreda koji predstavljaju model tj. relacije iz baze podataka.

```

@Entity
@Table(name = "racun")
@TypeDef(name = "pgsql_enum", typeClass = PostgreSQLEnumType.class)
public class Receipt {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_racun")
    private int id;

    @Column(name = "vrsta_racun", columnDefinition = "vrsta_racuna")
    @Enumerated(EnumType.STRING)
    @Type(type = "pgsql_enum")
    @NotNull(message = "Obavezno polje")
    private ReceiptType type;

    @Column(name = "iznos_racun")
    @NotNull(message = "Obavezno polje")
    @DecimalMin(value = "0", inclusive = false,
        message = "Mora biti veći od 0")
    @DecimalMax(value = "10000000",
        message = "Mora biti manji od 10 000 000")
    private Float value;

    @Column(name = "vrijeme_racun")
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME)
    @NotNull(message = "Obavezno polje")
    private Timestamp time;

    @Column(name = "napomena")
    private String description;

    @ManyToOne(cascade={CascadeType.DETACH, CascadeType.MERGE,
        CascadeType.PERSIST, CascadeType.REFRESH})
    @JoinColumn(name = "id_projekt")
    private Project projectReceipt;

    @ManyToOne(cascade={CascadeType.DETACH, CascadeType.MERGE,
        CascadeType.PERSIST, CascadeType.REFRESH})
    @JoinColumn(name = "id_okupljanje")
    private Gathering gatheringReceipt;

    public Receipt() {

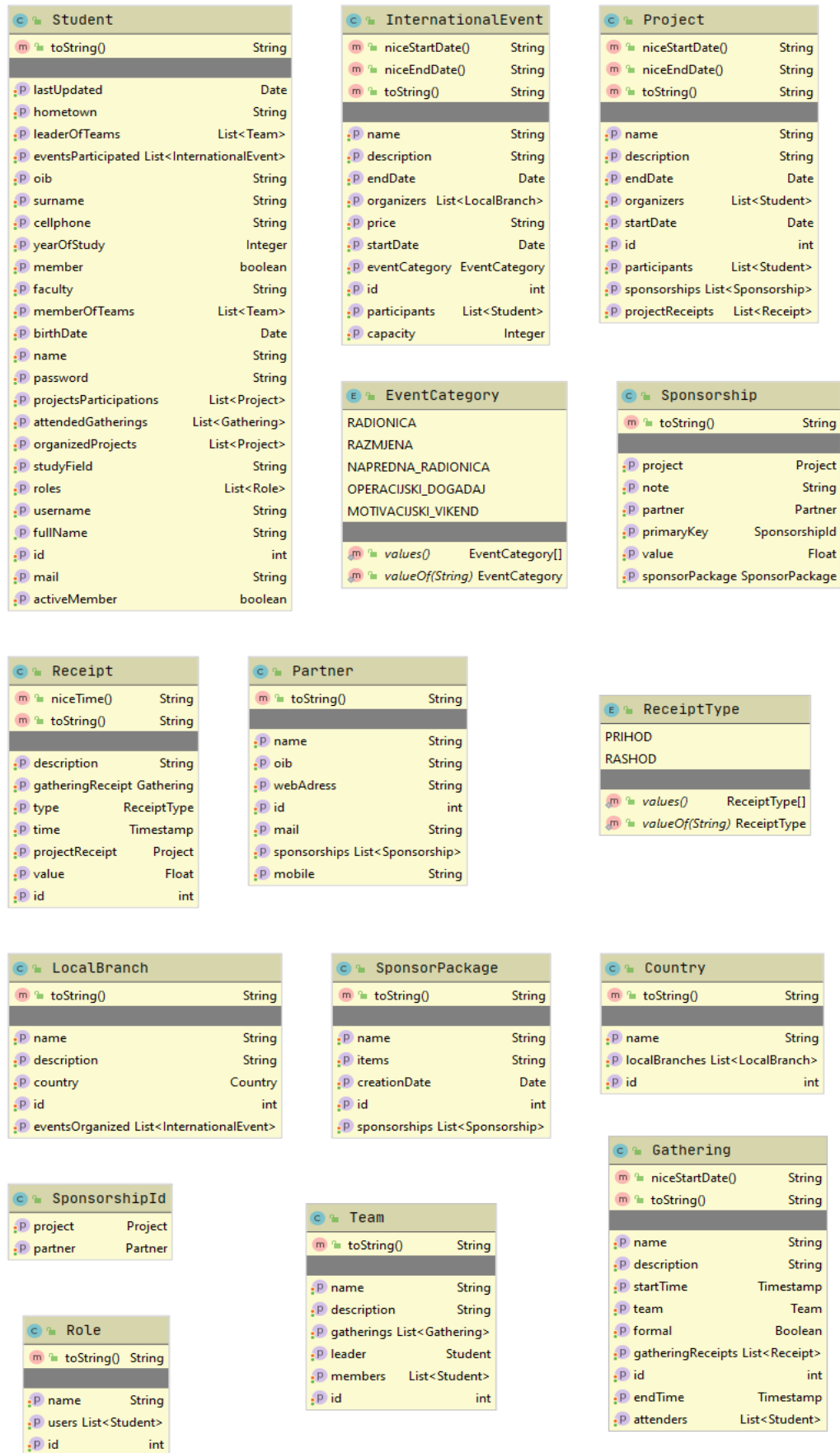
        this.time = new Timestamp(System.currentTimeMillis());
    }

    // Zbog čitljivosti izostavljena metode za postavljanje i dohvaćanje
    atributa (eng. getters and setters)

}

```

Kod 27. Isječak koda implementacije klase Receipt koja predstavlja relaciju RACUN



Powered by yFiles

Slika 5. Dijagram razreda koji predstavljaju model podataka

Apache Maven je alat za upravljanje projektom prilikom razvoja programske potpore. Pomoću njega je jednostavno dodati razne biblioteke u projekt iz središnjeg repozitorija biblioteka. Također, lako je projekt prenijeti u drugo razvojno okruženje koje podržava Maven. Središnja datoteka ove tehnologije je *pom.xml*. To je konfiguracijska datoteka u kojoj su opisane postavke i svojstva projekta te dodatci i biblioteke koje projekt koristi. Dodatci i biblioteke se automatski preuzimaju i omogućava se njihovo korištenje. Slijedi odsječak koda iz navedene datoteke.

```
<?xml version="1.0" encoding="UTF-8" ?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>hr.unizg.fer.sudec</groupId>
  <artifactId>association_management</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>

  <name>association_management Maven Webapp</name>
  <url>https://github.com/Svudec/association_management</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <springframework.version>5.2.6.RELEASE</springframework.version>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>

    <!-- Spring MVC support -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${springframework.version}</version>
    </dependency>

    <!-- Spring Transactions -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-tx</artifactId>
      <version>${springframework.version}</version>
    </dependency>
  </dependencies>
</project>
```

Kod 28. Isječak koda konfiguracijske datoteke pom.xml

4. Korisničke upute

Razne vrste korisnika imaju različiti prikaz i ovlaštenja u aplikaciji. Postoje 3 vrste računa:

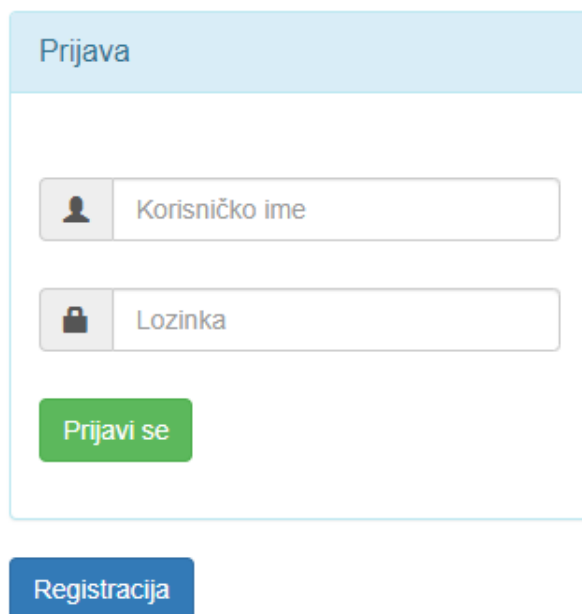
- Student
- Član upravnog odbora
- Administrator

Student može samo pregledavati određene podatke. Članovi upravnog odbora, uz pregledavanje, imaju mogućnost i izmjene svih podataka. Administratori imaju ovlasti mijenjanja dozvola korisničkim računima.

Svi korisnički računi su barem razine Student. Ostale razine su nadogradnja na ovlasti studenta.

4.1. Autorizacija korisnika

Aplikacija je namijenjena samo korisnicima koji posjeduju korisnički račun. Prilikom pristupa bilo kojoj stranici, neprijavljeni korisnik je preusmjeren na stranicu za prijavu. Na slici 6. je prikazana stranica za prijavu.

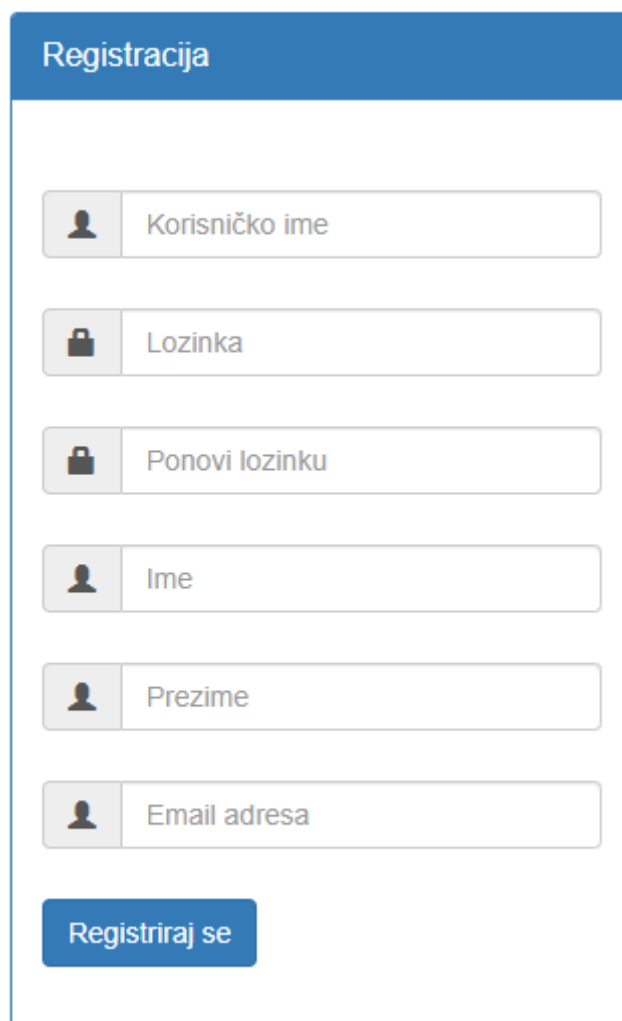


The image shows a login form titled "Prijava". It contains two input fields: the first is labeled "Korisničko ime" with a person icon, and the second is labeled "Lozinka" with a lock icon. Below these fields is a green button labeled "Prijavi se". At the bottom of the form is a blue button labeled "Registracija".

Slika 6. Prikaz stranice za prijavu

Stranica za prijavu od korisnika očekuje upis korisničkog imena i lozinke. Ukoliko korisnik još nema račun, može se registrirati pritiskom na tipku „Registracija“. Tada će biti preusmjeren na stranicu za registraciju čiji izgled je prikazan na slici 7.

Nakon uspješne prijave korisnik je preusmjeren na početnu stranicu.



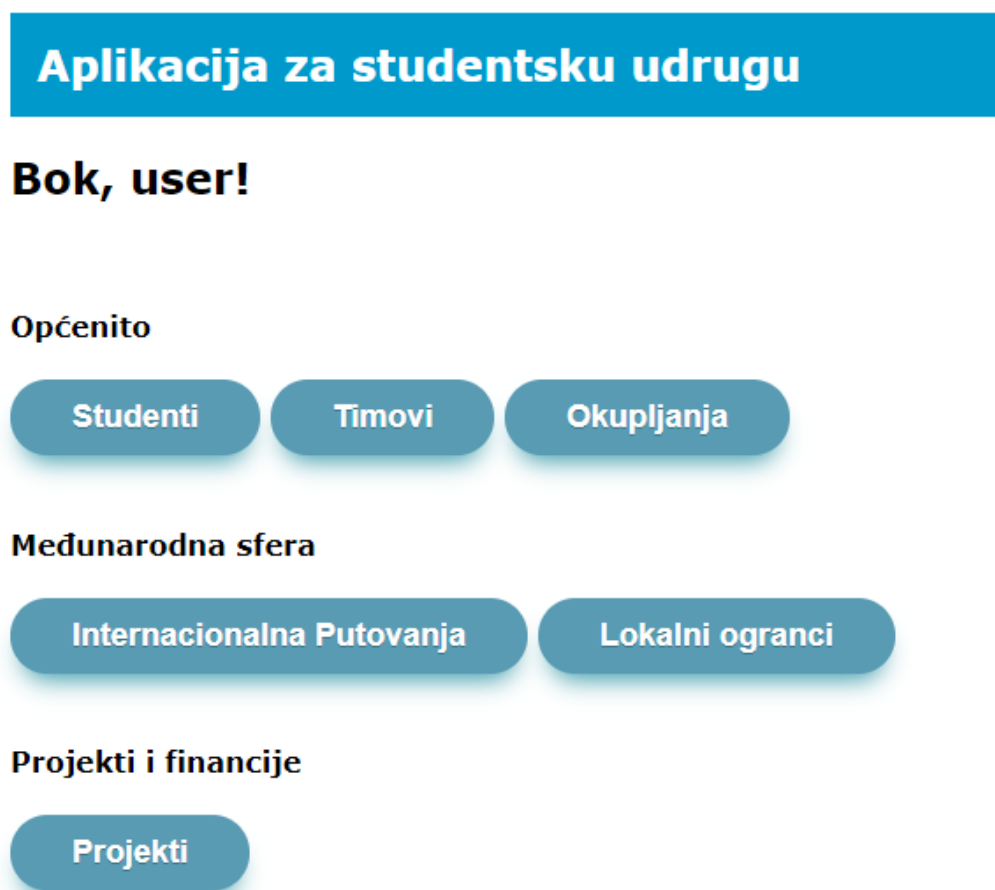
The image shows a registration form with a blue header bar containing the title "Registracija". Below the header, there are six input fields, each with a user icon on the left and a label: "Korisničko ime", "Lozinka", "Ponovi lozinku", "Ime", "Prezime", and "Email adresa". At the bottom of the form is a blue button with the text "Registriraj se".

Slika 7. Prikaz stranice za registraciju

Stranica za registraciju od korisnika zahtjeva upis osnovnih podataka za kreiranje računa. Ukoliko je neki od podataka neispravan ili upisano korisničko ime već postoji, stranica će prikazati grešku i registracija neće biti izvršena. Ukoliko je registracija uspješna, korisniku će to biti prikazano te će biti preusmjeren na stranicu za prijavu.

4.2. Student

Početni zaslon studenta je prikazan na slici 8. Postoje tri kategorije djelovanja udruge. Općenito djelovanje je vezano uz članove, timove i lokalna okupljanja. Međunarodnu sferu čine internacionalni događaji i lokalni ogranci udruge u drugim mjestima. Projekti su izdvojeni u treću kategoriju. Pritiskom na neku od tipki, otvara se izbornik tog područja.



Slika 8. Prikaz početne stranice studenta

4.2.1. Članovi

Pritiskom na tipku „Studenti“ u glavnom izborniku može se vidjeti popis svih studenata koji su sudjelovali u radu udruge. Za studente za koje postoji zapise o fakultetu na kojem studiraju i godini studija, oni se ispisuju. Također, ispisuje se broj okupljanja na kojima je student prisustvovao i broj putovanja koja je ostvario. Slika 9. prikazuje stranicu za pregled članova.

Aplikacija za studentsku udrugu

Ime i prezime	Fakultet	Godina studija	Broj okupljanja	Broj putovanja
Ringo Albertson	University of Lapland	3	1	2
Jammie Alfuso			2	2
Dionysus Alway			5	1
Georas Alywen			0	0
Boigle Antonoczyk			0	1
Kaiser Anyene	Sadat Institute of Higher Education	3	5	2
Michell Appleton			3	1
Brandon Atwell			2	1
Vitoria Avramovsky			2	2
Karly Bacher			1	2
Lynn Baert	Troy University	1	1	1
Beilul Barby			2	1
Fergus Battman			4	1
Ricky Baysting	Ural State Forestry Technical Academy	5	0	0
Wyatt Beneteau			1	1
Fredric Biddulph	Kuban State University of Technology	1	0	0
Balduin Bilofsky			2	2
Carson Blankenship	Siam University	2	2	0
Chad Bogays			1	4
Ekaterina Boleyn	Universidade de São Paulo	3	0	0

Slika 9. Prikaz stranice za pregled članova

4.2.2. Timovi

Pritiskom na tipku „Timovi“ u glavnom izborniku može se vidjeti popis svih timova koji djeluju u sklopu udruge. Prikazuje se naziv tima, ime voditelja i broj članova. Opisani prikaz vidljiv je na slici 10. Pored svakog tima postoji link pomoću kojeg se mogu vidjeti detalji tima. Na slici 11. je prikaz detalja tima.

Aplikacija za studentsku udrugu

Naziv	Voditelj	Broj članova	
Tim za odnose s javnošću	Benjie Pavlovsky	5	Detalji
Tim za prikupljanje financijskih sredstava	Bea Connerly	5	Detalji
Ljudski resursi	Karly Bacher	9	Detalji
Grant tim	Stefa Snar	6	Detalji
Tim za međunarodnu suradnju	Gino Camamill	5	Detalji
tincidunt congue turpis. In condimentum.	Rodolph Smeed	5	Detalji
Donec porttitor tellus non magna.	Carol-jean Crandon	3	Detalji
mi enim, condimentum eget, volutpat	Welsh McClelland	7	Detalji
risus. Quisque libero lacus, varius	Sofie Gladwish	6	Detalji
Aliquam erat volutpat. Nulla facilisis.	Jim Goulborn	9	Detalji

Slika 10. Prikaz stranice za pregled timova

Aplikacija za studentsku udrugu

Tim

Naziv:

Opis:

vitae, orci. Phasellus dapibus quam quis diam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Fusce aliquet magna a

Voditelj:

Slika 11. Prikaz stranice za pregled detalja tima

4.2.3. Okupljanja

Pritiskom na tipku „Okupljanja“ u glavnom izborniku može se vidjeti popis svih okupljanja udruge. Prikazuju se vrijeme i naziv okupljanja, broj prisutnih članova i ukupni troškovi na okupljanje. Opisani prikaz vidljiv je na slici 12. Pored svakog okupljanja postoji link pomoću kojeg se mogu vidjeti detalji okupljanja. Na slici 13. je prikaz detalja okupljanja.

Aplikacija za studentsku udrugu

Vrijeme	Naziv	Broj članova	Troškovi	
29-12-2019 17:37	sit amet diam	3	0.00	Detalji
22-12-2019 11:06	accumsan interdum libero	4	0.00	Detalji
08-12-2019 01:00	urna. Nunc quis	1	0.00	Detalji
06-12-2019 08:44	ac mattis ornare,	4	-343130.95	Detalji
18-11-2019 22:47	leo. Morbi neque	3	0.00	Detalji
18-11-2019 20:36	lectus pede et	5	0.00	Detalji
15-11-2019 17:01	vel, vulputate eu,	1	0.00	Detalji
15-11-2019 11:07	semper cursus. Integer	3	0.00	Detalji
11-11-2019 14:23	elit, pellentesque a,	4	0.00	Detalji
17-10-2019 16:43	erat, eget tincidunt	3	0.00	Detalji
14-10-2019 02:01	Suspendisse tristique neque	2	770802.81	Detalji
11-10-2019 03:18	euismod et, commodo	4	0.00	Detalji
01-10-2019 14:45	Aliquam auctor, velit	2	0.00	Detalji
20-09-2019 06:05	dictum sapien. Aenean	1	0.00	Detalji
09-09-2019 02:10	Donec felis orci,	3	0.00	Detalji
03-09-2019 11:27	egestas lacinia. Sed	4	0.00	Detalji
02-09-2019 00:38	libero est, congue	1	0.00	Detalji
23-08-2019 09:24	lectus ante dictum	1	-647126.19	Detalji
09-08-2019 09:36	vehicula. Pellentesque tincidunt	0	0.00	Detalji
02-08-2019 21:11	fringilla euismod enim.	1	0.00	Detalji
01-08-2019 07:54	consequat dolor vitae	4	746542.31	Detalji
29-07-2019 13:31	lacinia orci, consectetur	1	0.00	Detalji

Slika 12. Prikaz stranice za pregled okupljanja

Aplikacija za studentsku udrugu

Okupljanje

Naziv:	<input type="text" value="ac mattis ornare,"/>
Opis:	<div><div>Proin eget</div><div></div></div>
Početak:	<input type="text" value="06-Dec-2019 08:44:55"/>
Završetak:	<input type="text" value="18-Feb-2021 18:54:44"/>
Formalno:	<input checked="" type="checkbox"/>
Tim:	<input type="text" value="-----"/>

Računi

Vrijeme	Vrsta	Iznos	Opis
12-11-2017 10:30	-	555553.00	pede. Nunc sed orci lobortis augue
14-07-2019 01:54	+	212422.05	lacus. Quisque imperdiet, erat nonummy

Slika 13. Prikaz stranice za pregled detalja okupljanja

4.2.4. Internacionalni događaji

Pritiskom na tipku „Internacionalna putovanja“ u glavnom izborniku može se vidjeti popis svih internacionalnih događaja drugih lokalnih ogranaka udruge. Prikazuju se vrijeme, naziv, vrsta, ukupan broj sudionika i cijena participacije događaja. Opisani prikaz vidljiv je na slici 14. Pored svakog događaja postoji link pomoću kojeg se mogu vidjeti detalji događaja. Na slici 15. je prikaz detalja događaja.

Aplikacija za studentsku udrugu

Početak	Naziv	Vrsta	Broj sudionika	Cijena(EUR)	
24-04-2019	odio sagittis semper.	MOTIVACIJSKI_VIKEND	90	111.00	Detalji
18-04-2019	Curabitur massa. Vestibulum	MOTIVACIJSKI_VIKEND	19	73.00	Detalji
06-04-2019	sodales at, velit.	RADIONICA	19	0.00	Detalji
23-03-2019	sed, sapien. Nunc	MOTIVACIJSKI_VIKEND	90	162.00	Detalji
13-03-2019	tempus, lorem fringilla	OPERACIJSKI_DOGADAJ	75	0.00	Detalji
05-03-2019	Nullam feugiat placerat	RADIONICA	70	0.00	Detalji
25-01-2019	volutpat. Nulla dignissim.	MOTIVACIJSKI_VIKEND	88	197.00	Detalji
16-01-2019	Cum sociis natoque	NAPREDNA_RADIONICA	29	0.00	Detalji
14-01-2019	Sed malesuada augue	RADIONICA	35	0.00	Detalji
13-01-2019	orci sem eget	OPERACIJSKI_DOGADAJ	64	0.00	Detalji
10-01-2019	amet, consectetur adipiscing	OPERACIJSKI_DOGADAJ	69	0.00	Detalji
29-12-2018	eu dolor egestas	NAPREDNA_RADIONICA	83	0.00	Detalji
27-12-2018	tempor arcu. Vestibulum	OPERACIJSKI_DOGADAJ	19	0.00	Detalji
01-12-2018	Mauris vestibulum, neque	OPERACIJSKI_DOGADAJ	26	0.00	Detalji
08-11-2018	mollis. Duis sit	NAPREDNA_RADIONICA	74	0.00	Detalji
29-10-2018	tincidunt adipiscing. Mauris	RAZMJENA	60	141.00	Detalji
28-09-2018	mauris elit, dictum	RADIONICA	67	0.00	Detalji
18-09-2018	adipiscing ligula. Aenean	RAZMJENA	46	221.00	Detalji
16-08-2018	eu metus. In	MOTIVACIJSKI_VIKEND	55	177.00	Detalji
17-07-2018	ornare placerat, orci	NAPREDNA_RADIONICA	74	0.00	Detalji
26-06-2018	nec, diam. Duis	MOTIVACIJSKI_VIKEND	95	104.00	Detalji
15-06-2018	dictum augue malesuada	NAPREDNA_RADIONICA	98	0.00	Detalji

Slika 14. Prikaz stranice za pregled međunarodnih događaja

Aplikacija za studentsku udrugu

tincidunt adipiscing. Mauris

Opis: adipiscing elit. Aliquam auctor, velit eget laoreet posuere, enim nisl elementum purus, accumsan interdum libero dui nec

Početak: 29-10-2018

Završetak: 18-05-2019

Ukupno
sudionika: 60

Cijena: 141.00 €

Vrsta: RAZMJENA

Organizator(i): Herfelingen

Naši sudionici

Ime i prezime	Fakultet	Godina studija	Broj okupljanja	Broj putovanja
Thain Troke			4	2
Murdoch			2	2
Bonallack				
Balduin Bilofsky			2	2

Slika 15. Prikaz stranice za pregled detalja međunarodnog događaja

4.2.5. Lokalni ogranci udruge

Pritiskom na tipku „Lokalni ogranci“ u glavnom izborniku može se vidjeti popis svih lokalnih ogranka udruge, grupiran prema državama. Prikazuju se država, ogranak i broj organiziranih internacionalnih događaja. Opisani prikaz vidljiv je na slici 16. Pored svakog ogranka postoji link pomoću kojeg se može vidjeti popis svih događaja koje je taj ogranak organizirao. Navedena stranica je slična popisu internacionalnih događaja iz poglavlja 4.2.4.

Aplikacija za studentsku udrugu			
Država	Ogranak	Organizirano događanja	
Albania	Glabaïs	3	Događanja
	Fresia Nuova	2	Događanja
	Great Falls	4	Događanja
Armenia	Portree	2	Događanja
	Metairie	1	Događanja
	Champdani	2	Događanja
Austria	Dubna	1	Događanja
	Zignago	2	Događanja
	Lethbridge	0	Događanja
Azerbaijan	Peumo	2	Događanja
	Buguma	5	Događanja
	Cerchio	1	Događanja
Belarus	Sukkur	0	Događanja
	Pietragalla	3	Događanja
	Athens	1	Događanja
Belgium	San Martino in Pensilis	1	Događanja
	Blevio	0	Događanja
	Oyace	2	Događanja
Bosnia and Herzegovina	Bard	0	Događanja
	Castelmezzano	2	Događanja
	Hexham	0	Događanja
Bulgaria	Bressoux	0	Događanja

Slika 16. Prikaz stranice za pregled lokalnih ogranka

4.2.6. Projekti

Pritiskom na tipku „Projekti“ u glavnom izborniku može se vidjeti popis svih projekata koje je udruga organizirala. Prikazuju se datum i naziv projekta, broj sudionika, broj sponzora te ukupna zarada. Opisani prikaz vidljiv je na slici 17.

Aplikacija za studentsku udrugu				
Datum	Naziv	Broj sudionika	Broj sponzorstava	Zarada
20-03-2018	risus odio, auctor	3	4	1178336.58
07-02-2018	Mauris eu turpis.	4	4	647943.01
31-01-2018	vitae dolor. Donec	4	4	81102.71
24-01-2018	id risus quis	7	4	38222.73
30-11-2017	risus. Donec egestas.	5	4	755540.21
30-10-2017	Sed auctor odio	5	4	140410.47
25-09-2017	nonummy ac, feugiat	9	3	-474829.77
02-09-2017	orci, adipiscing non,	6	4	120221.57
18-08-2017	sit amet diam	4	4	142490.20
15-08-2017	Praesent luctus. Curabitur	7	4	260486.56
24-07-2017	eu neque pellentesque	5	4	125673.57
03-06-2017	eu elit. Nulla	5	3	101596.42
09-05-2017	purus, accumsan interdum	6	4	108017.53
09-05-2017	pede, malesuada vel,	6	4	112650.08
06-05-2017	gravida non, sollicitudin	3	4	-239940.27
04-05-2017	purus. Nullam scelerisque	8	4	138532.39
24-04-2017	ultrices ornare, elit	7	4	331279.25
20-04-2017	eu metus. In	3	4	-521408.50
19-03-2017	non enim commodo	3	4	54884.92
15-03-2017	Mauris ut quam	2	4	74624.74
13-03-2017	euismod enim. Etiam	4	4	-633043.50
22-11-2016	egestas. Aliquam fringilla	3	4	-391063.08

Slika 17. Prikaz stranice za pregled projekata

4.3. Član upravnog odbora

Početni zaslon člana upravnog odbora je prikazan na slici 18. Sličan je početnom zaslonu studenta, no ima još i tipku „Financije“. Pritiskom na neku od tipki, otvara se izbornik tog područja. Pregled podataka je isti kao i kod studenta, te taj dio neće biti ponovno opisan.



Slika 18. Prikaz početne stranice člana upravnog odbora

Gotovo na svim stranicama, član upravnog odbora može dodavati nove zapise te izmjenjivati i brisati postojeće. Na slici 19. prikazan je popis studenata, a na slici 20. forma za kreiranje novog studenta.

Aplikacija za studentsku udrugu					
Dodaj studenta					
Ime i prezime	Fakultet	Godina studija	Broj okupljanja	Broj putovanja	
Ringo Albertson	University of Lapland	3	1	2	Otvori profil Ukloni
Jammie Alfuso			2	2	Otvori profil Ukloni
Dionysus Alway			5	1	Otvori profil Ukloni
Georas Alywen			0	0	Otvori profil Ukloni
Boigie Antonoczky			0	1	Otvori profil Ukloni
Kaiser Anyene			5	2	Otvori profil Ukloni
Michell Appleton			3	1	Otvori profil Ukloni
Brandon Atwell	Sadat Institute of Higher Education	3	2	1	Otvori profil Ukloni
Vitoria Avramovsky			2	2	Otvori profil Ukloni
Karly Bacher			1	2	Otvori profil Ukloni

Slika 19. Prikaz stranice za pregled studenata (upravni odbor)

Aplikacija za studentsku udrugu

Student

Ime:	<input type="text"/>
Prezime:	<input type="text"/>
Email adresa:	<input type="text"/>
Mobitel:	<input type="text"/>
OIB:	<input type="text"/>
Datum rođenja:	<input type="text" value="dd-----yyyy"/> <input type="button" value="📅"/>
Mjesto prebivališta:	<input type="text"/>
Fakultet:	<input type="text"/>
Godina studija:	<input type="text"/>
Smjer studija:	<input type="text"/>
Student je član udruge:	<input type="checkbox"/>
Student je AKTIVAN član udruge:	<input type="checkbox"/>
<input type="button" value="Spremi"/>	

Slika 20. Prikaz stranice za dodavanje novog studenta

Pritiskom na tipku „Financije“ otvara se pregled svih računa udruge, te je vidljivo i trenutno stanje računa. Na slici 21. je prikaz spomenutog pregleda.

Aplikacija za studentsku udrugu

Trenutno stanje računa: **2,468,384.19**

Vrijeme	Vrsta	Iznos	Opis
03-06-2020 15:47	+	4686.67	Račun od sponzorstva; SPONZOR: Wikivu; PROJEKT: Mauris eu turplis.
03-06-2020 15:47	+	11233.80	Račun od sponzorstva; SPONZOR: Edgepulse; PROJEKT: tellus non magna.
03-06-2020 15:47	+	9921.78	Račun od sponzorstva; SPONZOR: Livefish; PROJEKT: ultricies ornare, elit
03-06-2020 15:47	+	33303.77	Račun od sponzorstva; SPONZOR: Aibox; PROJEKT: orci, adipiscing non,
03-06-2020 15:47	+	47141.02	Račun od sponzorstva; SPONZOR: Aibox; PROJEKT: purus, accumsan interdum
03-06-2020 15:47	+	23356.75	Račun od sponzorstva; SPONZOR: Livefish; PROJEKT: Mauris eu turplis.
03-06-2020 15:47	+	37769.69	Račun od sponzorstva; SPONZOR: Skimia; PROJEKT: ut erat. Sed
03-06-2020 15:47	+	20214.97	Račun od sponzorstva; SPONZOR: Digitube; PROJEKT: mi. Duis risus
03-06-2020 15:47	+	10776.95	Račun od sponzorstva; SPONZOR: Blognation; PROJEKT: neque non quam.
03-06-2020 15:47	+	8021.21	Račun od sponzorstva; SPONZOR: Skynoodle; PROJEKT: mollis. Integer tincidunt
03-06-2020 15:47	+	30275.87	Račun od sponzorstva; SPONZOR: Devpoint; PROJEKT: purus, accumsan interdum
03-06-2020 15:47	+	6342.28	Račun od sponzorstva; SPONZOR: Dynava; PROJEKT: Praesent luctus. Curabitur
03-06-2020 15:47	+	11462.40	Račun od sponzorstva; SPONZOR: Photojam; PROJEKT: vitae dolor. Donec
03-06-2020 15:47	+	2431.89	Račun od sponzorstva; SPONZOR: Skimia; PROJEKT: risus odio, auctor

Slika 21. Prikaz stranice za pregled financija

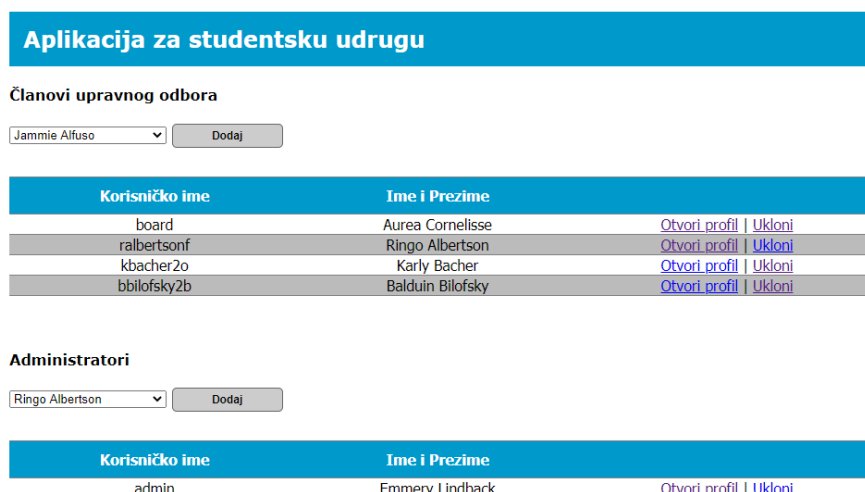
4.4. Administrator

Početni zaslon administratora je prikazan na slici 22. Sličan je početnom zaslonu studenta, no ima još i tipku „Administriranje“. Pritiskom na neku od tipki, otvara se izbornik tog područja. Pregled podataka je isti kao i kod studenta, te taj dio neće biti ponovno opisan.



Slika 22. Prikaz početne stranice administratora

Pritiskom na tipku „Administriranje“ otvara se stranica s prikazom trenutnih članova upravnog odbora i administratora. Na stranici je moguće dodijeljivati ovlaštenja korisničkim računima i uklanjati postojeća. Prikaz opisane stranice nalazi se na slici 23.



Slika 23. Prikaz stranice za administriranje

Zaključak

Cilj ovoga rada bio je definirati, oblikovati i implementirati programsku potporu za vođenje evidencije o radu studentske udruge. Trenutno, velika većina udruga ovakve podatke vodi u raznim tablicama, formama i dokumentima. To je dugoročno neodrživo jer se dio starijih podataka gotovo sigurno izgubi. Također, teško je uvidjeti neke trendove kada podatci nisu adekvatno organizirani.

Baza podataka je implementirana u PostgreSQL tehnologiji, a web aplikacija koristeći programski jezik Java te Spring framework i Hibernate framework. Smatram da je cilj generalno ostvaren. Moguće je čuvati sve potrebne podatke putem aplikacije i baze podataka.

Nedostatak je pomalo zastarjelo i ponegdje neintuitivno grafičko sučelje. Također, performanse nekih dijelova aplikacije nisu najbolje.

Projekt bi se mogao nadograditi implementacijom modernijeg i jednostavnijeg sučelja te poboljšanjem sigurnosti i performansi. Također, moguća je integracija s nekom od društvenih mreža ili čak sveučilišnom informatičkom infrastrukturom.

Literatura

- [1] Manger, R. *Baze podataka*, Element, 2. izdanje, 2014.
- [2] Darby, C., *Spring & Hibernate for Beginners (includes Spring Boot)*, Udemy (svibanj 2020), Poveznica: <https://www.udemy.com/course/spring-hibernate-tutorial/>; pristupljeno 02. lipnja 2020.
- [3] *Dependency injection (inversion of control)*, Učim programiranje (veljača 2013.), Poveznica: <https://www.ucim-programiranje.com/2013/02/dependency-injection-inversion-of-control/>; pristupljeno 03. lipnja 2020.
- [4] *CSS Reference*, w3schools.com, Poveznica: <https://www.w3schools.com/cssref/default.asp>; pristupljeno 1. lipnja 2020.
- [5] Ha Minh, N., *Hibernate Many-to-Many Association with Extra Columns in Join Table Example*, Code Java (svibanj 2020.), Poveznica: <https://www.codejava.net/frameworks/hibernate/hibernate-many-to-many-association-with-extra-columns-in-join-table-example>; pristupljeno 25. svibnja 2020.
- [6] Nastavni materijali iz kolegija Baze podataka, Fakultet elektrotehnike i računarstva, ožujak 2018.
- [7] Nastavni materijali iz kolegija Oblikovanje programske potpore, Fakultet elektrotehnike i računarstva, 2018./2019.
- [8] *Hrvatski opći leksikon*, Leksikografski zavod Miroslav Krleža, Zagreb, 1996.

Sažetak

Baza podataka i web aplikacija za udrugu

U ovom radu opisuje se oblikovanje programske potpore za vođenje evidencije rada studentske udruge. Programska potpora se sastoji od PostgreSQL baze podataka i web aplikacije implementirane programskim jezikom Java uz korištenje Spring framework i Hibernate framework, kao i HTML, i CSS.

U aplikaciji se vodi evidencija o članovima, okupljanjima, radu timova, internacionalnim događajima, organiziranim projektima i financijama udruge.

Rad sadrži opis problema, model podataka, implementaciju i korisničke upute za aplikaciju.

Ključne riječi: udruga, baza podataka, PostgreSql, Java, Spring, Hibernate, relacijski model

Summary

Database and web application for the association

This paper describes the design of software for keeping records of the student association. Software consists of a PostgreSQL database and a web application implemented in the Java programming language using the Spring framework and Hibernate framework, as well as HTML and CSS.

The application keeps records of members, gatherings, teams, international events, projects and the finances of the association.

The paper contains a description of the problem area, data model, implementation and user guide for the application.

Keywords: association, database, PostgreSQL, Java, Spring, Hibernate, relational model