

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2902

**Razvoj informacijskog sustava za
potporu organizaciji događanja s
velikim brojem sudionika
korištenjem programskog okvira
za ubrzani razvoj aplikacija**

Fran Borčić

Zagreb, lipanj 2022.

DIPLOMSKI ZADATAK br. 2902

Pristupnik: **Fran Borčić (0036498432)**

Studij: Računarstvo

Profil: Programsko inženjerstvo i informacijski sustavi

Mentor: doc. dr. sc. Slaven Zakošek

Zadatak: **Razvoj informacijskog sustava za potporu organizaciji događanja s velikim brojem sudionika korištenjem programskog okvira za ubrzani razvoj aplikacija**

Opis zadatka:

Načiniti pregled funkcionalnosti i primjenjivosti postojećih ERP rješenja otvorenog koda u području organizacije događanja s velikim brojem sudionika. Proučiti i dokumentirati poslovne procese za pripremu i upravljanje događanjima s velikim brojem sudionika na primjeru organizacije studentskog natjecanja STEM Games. Korištenjem odabranog programskog okvira za ubrzani razvoj aplikacija (engl. Rapid Application Development) izgraditi inicijalnu, ali funkcionalnu inačicu sustava za potporu organizaciji i upravljanju natjecanjem. Razvijeni sustav treba, između ostalog, omogućiti evidentiranje podataka o sudionicima, inventaru i suradnji s velikim brojem uključenih strana, npr. institucijama sudionicama, medijskim partnerima, sponzorima i kooperantima. Uspješnost sustava ispitati u praksi te dokumentirati rezultate. Na temelju dobivenih rezultata ocijeniti opravdanost razvoja ovakvog sustava u odnosu na korištenje postojećih ERP rješenja otvorenog koda.

Rok za predaju rada: 27. lipnja 2022.

Zahvaljujem mentoru profesoru Slavenu Zakošku na pomoći pri izradi ovog rada.

SADRŽAJ

1. Uvod	1
2. Primjer strukture organizacije	3
2.1. Natjecanje STEM Games	3
2.2. Hijerarhijska struktura organizacije	4
2.3. Zadaci voditelja organizacije	5
2.4. Zadaci studentskih predstavnika	6
2.5. Zadaci operativnog tima	6
2.5.1. Nabava i upravljanje inventarom	7
2.5.2. Lokalni prijevoz i prijevoz organizacije	7
2.5.3. Upravljanje sudionicima	8
2.5.4. Dodatni program, raspored i dodatni volonteri	8
2.6. Zadaci tima za partnerstva	8
2.7. Zadaci tima za marketing i komunikacije te podtimova	9
2.8. Timovi za natjecanja u sportu, znanju i računalnim igrama	9
2.9. Ostale potrebe	9
3. Podaci organizacije za pohranu u IS	11
3.1. Podaci o osobama	11
3.2. Korisnici sustava i uloge	12
3.3. Podaci o organizacijama i institucijama	12
3.4. Podaci o projektima	13
3.5. Podaci o sudjelovanju obrazovne institucije	13
3.6. Podaci o inventaru	13
3.7. Podaci o dokumentu	14
3.8. Podaci o sponzoru	14
3.9. Podaci o sudjelovanju natjecatelja	14
3.10. Podaci o sudjelovanju volontera	15

3.11. Podaci o računima	15
3.12. Podaci o vozilima	15
3.13. Podaci o budžetu	16
3.14. Podaci o kooperantima	16
3.15. Pristupni podaci za mrežne usluge	16
3.16. Podaci o medijskom partneru	16
4. Prikladnost <i>ERP</i> sustava otvorenog koda	17
5. Rješenje primjenom <i>RAD</i> programskog okvira	19
5.1. Entiteti podatkovnog modela	20
5.1.1. Entitet <i>Project</i>	20
5.1.2. Entitet <i>Organization</i>	20
5.1.3. Entitet <i>Person</i>	21
5.1.4. Entitet <i>EduInstitution</i>	21
5.1.5. Entitet <i>OrganizationTeam</i>	22
5.1.6. Entitet <i>ProjectOrganizationTeam</i>	22
5.1.7. Entitet <i>OrganizationVolunteer</i>	23
5.1.8. Entitet <i>InstitutionParticipation</i>	23
5.1.9. Entitet <i>InstitutionRepresentation</i>	23
5.1.10. Entitet <i>Location</i>	24
5.1.11. Entitet <i>InventoryItem</i>	24
5.1.12. Entitet <i>Sponsor</i>	25
5.1.13. Entitet <i>SponsorComments</i>	26
5.1.14. Entitet <i>SponsorCategory</i>	26
5.1.15. Entitet <i>SponsorResponsivenessLevel</i>	26
5.1.16. Entitet <i>Subcontractor</i>	27
5.1.17. Entitet <i>SubcontractorComments</i>	27
5.1.18. Entitet <i>PhysicalDocument</i>	27
5.1.19. Entitet <i>DigitalDocument</i>	28
5.1.20. Entitet <i>InvoiceOutgoing</i>	28
5.1.21. Entitet <i>InvoiceOutgoingItem</i>	29
5.1.22. Entitet <i>InvoiceIncoming</i>	29
5.1.23. Entitet <i>Income</i>	30
5.1.24. Entitet <i>Expense</i>	31
5.1.25. Entitet <i>PlannedIncome</i>	31

5.1.26. Entitet <i>PlannedExpense</i>	31
5.1.27. Entitet <i>Contract</i>	32
5.1.28. Entitet <i>OrganizationContact</i>	32
5.1.29. Entitet <i>ParticipantInfo</i>	33
5.1.30. Entitet <i>ParticipationCategory</i>	34
5.1.31. Entitet <i>Vehicle</i>	34
5.1.32. Entitet <i>VehicleReservation</i>	34
5.1.33. Entitet <i>ToolCredentials</i>	35
5.1.34. Entiteti <i>User</i> , <i>Permission</i> i <i>Role</i>	35
5.2. Implementacija pomoću <i>Flask-Appbuilder</i> programskog okvira	36
5.2.1. Programska okolina	37
5.2.2. Definicija modela	38
5.2.3. Definicija korisničkog sučelja	40
5.2.4. Filtriranje i kontrola pristupa	42
5.3. Korištenje aplikacije	44
5.3.1. Glavno sučelje	44
5.3.2. Upravljanje korisnicima, dozvolama i ulogama	45
5.3.3. Upravljanje podacima	47
5.4. Nedostaci i prostor za nadogradnju	50
6. Zaključak	52
Literatura	53
A. Struktura direktorija razvijene aplikacije	54
B. Izbornici u razvijenoj aplikaciji	56

1. Uvod

Cilj ovog rada je ocijeniti mogućnost i isplativost implementacije informacijskog sustava u rad studentskih organizacija koje se bave organizacijom studentskih natjecanja s velikim brojem sudionika. Problemu će se pristupiti s dva različita aspekta – analizom prikladnosti postojećih rješenja i razvojem osnovnog, ali primjenjivog prilagođenog rješenja koristeći programski okvir za ubrzani razvoj aplikacija (engl. *RAD*), a vodeći se konkretnim primjerom *Udruge za studentska natjecanja STEM Games*.

Tijekom posljednjih nekoliko desetljeća svjedočimo sve većoj prisutnosti informacijskih sustava u poslovanju komercijalnih organizacija, većinom u vidu radnih okvira poznatih kao *ERP* sustavi. Pojam *ERP rješenje* (engl. *enterprise resource planning*) obuhvaća veliki broj informacijskih sustava različitih vrsta i primjena, no prema definicijama navedenima u (Rashid M. A. (2002)) taj se pojam odnosi na sustave koje karakterizira modularnost te centralizirana pohrana svih podataka vezanih uz poslovanje jedne organizacije.

Većina takvih sustava danas je realizirana kao skup gotovih modula oblikovanih prema generičkim potrebama većeg broja komercijalnih organizacija (Bradford (2014)). S obzirom na broj poslovnih procesa prisutnih u svakoj takvoj organizaciji, takva arhitektura omogućuje ubranu implementaciju većeg dijela informacijskog sustava i ponovnu uporabu novorazvijenih modula u organizacijama sličnog područja rada.

Logična je posljedica spomenute arhitekture da je uvođenje *ERP* sustava u poslovanje to jeftinije i jednostavnije što je više poslovnih procesa organizacije u koju se uvodi prisutno i u drugim organizacijama. Organizacije koje se bave rijetkim i specifičnim zadacima ograničene su u broju dostupnih rješenja kojima mogu digitalizirati svoje poslovanje pa je slijedom toga i isplativost njihove informatizacije znatno manja.

Slučaj koji će u ovom radu biti proučen, osim zbog svojih poslovnih procesa, specifičan je i prema svojim ciljevima – organizacija studentskih natjecanja na području RH tradicionalno je neprofitnog karaktera, a iza takvih događaja stoje studentske udruge ili udruge građana čiji su članovi volonteri. Iz tog razloga, potencijalni benefiti uvođenja informacijskog sustava u tom području očitovali bi se prije svega u jednostavnosti

obavljanja volonterskog rada, a u znatno manjoj mjeri kroz financijske pokazatelje. Također, takve organizacije uglavnom ne mogu u svojim budžetima opravdati profesionalnu izradu informacijskog sustava, ali su im dostupni ljudski resursi u obliku studenata volontera.

Eventualni informacijski sustavi koji bi bili prikladni za takve organizacije, bili bi osmišljeni tako da se studenti volonteri uz minimalnu edukaciju mogu brinuti o uvođenju i održavanju takvih sustava. S obzirom na sličnosti u formatu različitih studentskih natjecanja na području RH, kroz ovaj rad nastojat će se pronaći rješenje koje bi olakšalo organizaciju svih takvih događanja, a čija primjena bi zahtijevala što manje tehničkog znanja.

Takvo što moglo bi se postići primjenom ERP rješenja otvorenog koda i prethodnom pripremom njihove konfiguracije za konkretni zadatak organizacije. ERP rješenja otvorenog koda za neprofitne organizacije postoje te će biti razmotrena u nastavku rada, no s obzirom na činjenicu da se u ovom slučaju radi o vrlo specifičnom tipu neprofitne organizacije takva rješenja ne mogu pokriti sve njene potrebe.

Bolje prilagođeno rješenje, ali tehnički zahtjevnije za održavanje, moglo bi se ostvariti korištenjem programskih okvira za ubrzani razvoj aplikacija (engl. *RAD*). Takvi programski okviri automatiziraju izradu čestih elemenata korisničkog sučelja prema vezanom podatkovnom modelu te time omogućuju brzu izradu programske potpore čiji je glavni zadatak pohrana podataka. Fokus ovog rada upravo je razvoj ovakvog rješenja, ocjena njegove primjenjivosti te potencijal za primjenu u većem broju organizacija.

U poglavlju koje slijedi bit će opisane organizacijske strukture *Udruge za studentska natjecanja STEM Games* te njihovi zadaci prema podacima iz godišnjih izvještaja različitih organizacijskih timova koji su sudjelovali na prethodnim projektima te udruge. Ti će zadaci, odnosno poslovni procesi koje oni uključuju, kasnije poslužiti za izolaciju zahtjeva na informacijski sustav koji bi pružao podršku organizaciji studentskih natjecanja. Zatim će biti razmotrena dostupna ERP rješenja otvorenog koda te broj zahtjeva koji bi se njihovom primjenom zadovoljio, nakon čega će biti opisano rješenje ostvareno koristeći RAD programski okvir. Naposljetku, razvijeno rješenje bit će empirički ocijenjeno prema iskustvima unutar spomenutog STEM Games organizacijskog tima te će se sažeti zaključci o primjenjivosti informacijskih sustava u organizacijama tog tipa.

2. Primjer strukture organizacije

U ovom poglavlju, na primjeru *Udruge za studentska natjecanja STEM Games* čiji je glavni projekt organizacija godišnjeg natjecanja za studente STEM fakulteta pod imenom STEM Games, bit će opisane temeljne procedure potrebne za organizaciju takvog događaja. Spomenuta organizacija, koju će se dalje u radu iz praktičnih razloga zvati *organizacija STEM Games*, poslužit će kao primjer jer je autoru koji je u trenutku pisanja rada na njenom čelu dostupan veliki broj podataka o dosadašnjim projektima udruge.

Zbog složenosti strukture organizacija ove vrste, praktičnije ju je opisati na konkretnom primjeru organizacije koja se njome bavi. Dalje će se, međutim, nastojati odvojiti koncepte proizašle iz opisa koji slijedi od konkretne STEM Games organizacije, u skladu s nastojanjem da rezultati rada budu primjenjivi na sve slične organizacije.

U svrhu lakšeg razumijevanja organizacije STEM Games, valja prvo ukratko opisati događaj kojim se bavi.

2.1. Natjecanje STEM Games

Organizacija STEM Games nastala je i djeluje s ciljem da se studentima STEM fakulteta omogući događanje na kojem se sudjelovanjem na natjecateljskim aktivnostima mogu povezati s kolegama s drugih institucija te predstavnicima industrije u njihovoj budućoj struci. Samo natjecanje osmišljeno je po modelu već postojećih studentskih događanja u regiji, uz određene prilagodbe vezane specifično uz STEM područje i suradnju s industrijom.

Organizacija jednom godišnje organizira natjecanje na kojem okuplja između tisuću i dvije sudionika, odnosno petnaest do dvadeset visokoobrazovnih institucija. Studenti sudionici, prema kategorijama dostupnim u trenutku pisanja ovog rada, natječu se u rješavanju problemskih zadataka STEM područja, sportskim disciplinama ili računalnim igrama. Uz to, na prijedlog volontera organizira se dodatni program u

obliku neslužbenih natjecanja, predavanja, koncertnih događanja i slično.

Središnji element ovog konkretnog natjecanja je tzv. *natjecanje u znanju* u kojem se timovi sudionika natječu u primjeni znanja stečenog na svojim fakultetima. Radi se o četiri odvojena natjecanja u kojima se natječu timovi studenata prirodnoslovnog, tehnološkog, inženjerskog ili matematičkog usmjerenja, rješavajući probleme koji se u suradnji s predstavnicima industrije modeliraju prema problemima s kojima će se sresti na budućem radnom mjestu.¹

Glavne zadaće organizacije STEM Games uključuju omogućavanje provedbe svih natjecateljskih elemenata, logističko i financijsko planiranje, prikupljanje sredstava, pronalazak volontera, komunikaciju s institucijama sudionicama i promotivne aktivnosti.

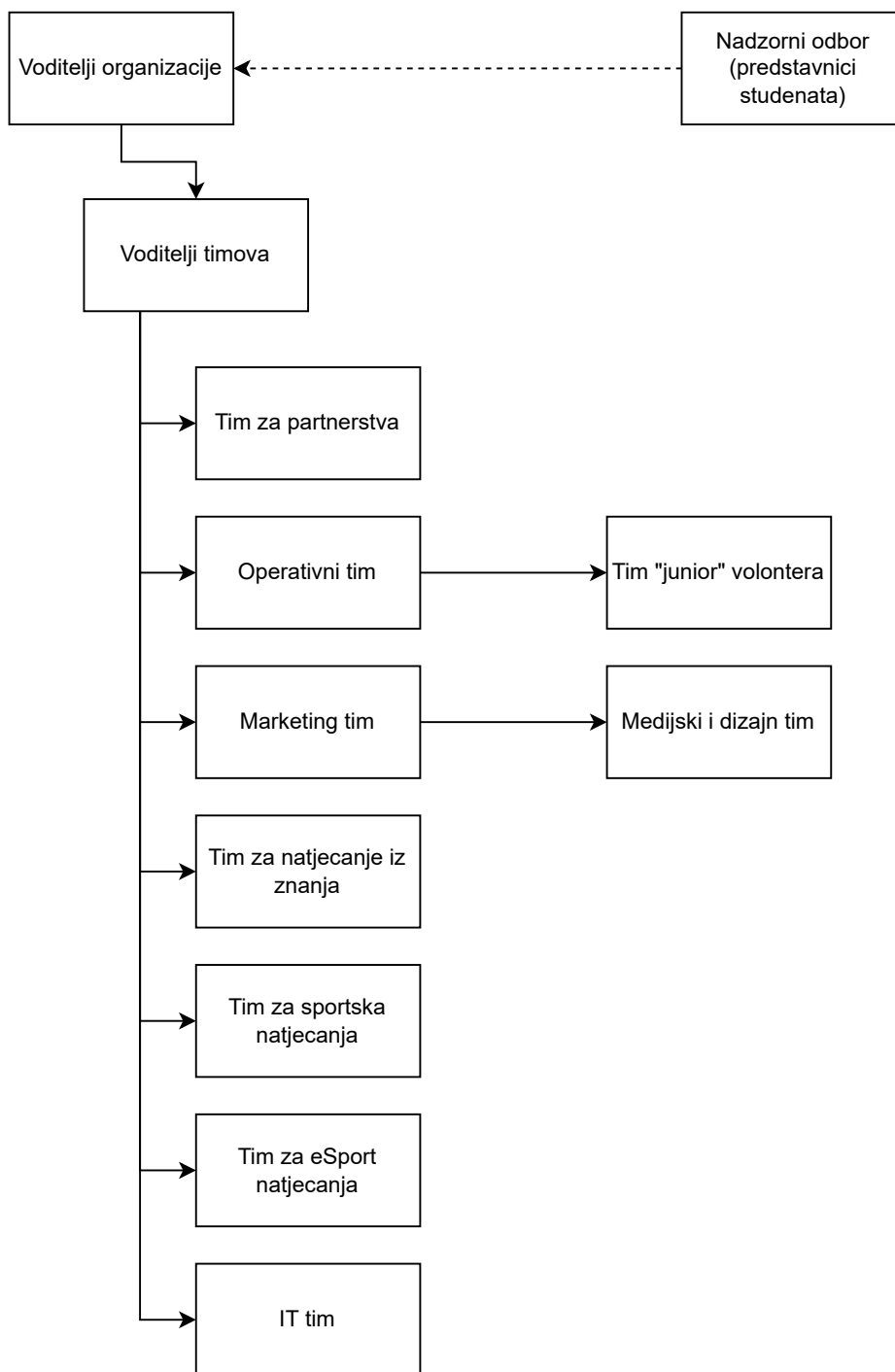
2.2. Hijerarhijska struktura organizacije

Zbog velikog opsega događanja, u organizaciji dosadašnjih izdanja svake je godine dosad sudjelovalo oko 150 volontera. Kako bi se organizacijski tim te veličine uspješno koordinirao, organizacija je ustrojena hijerarhijski te dijeli volontere u različite timove ovisno o zadatku kojim se bave.

Na slici 2.1 prikazana je podjela organizacije po timovima čiji će zadaci biti opisani u nastavku. Cjelokupnu organizaciju koordiniraju dva *voditelja organizacije* uz pomoć voditelja pojedinih organizacijskih timova, u kojima je također prisutan interni ustroj. Detalji ustroja timova za svrhe ovog rada nisu relevantni, no gruba podjela organizacije prikazana je kako bi se u nastavku organizacijski zadaci mogli smisleno podijeliti po timovima koji ih izvršavaju.

U potpoglavljima koja slijede po timovima će biti opisani oni njihovi zadaci u čijoj provedbi potencijalni informacijski sustav može biti koristan.

¹Taj element omogućuje financiranje organizacije kroz sponzorsku suradnju; u smislu ovog rada, bitno je uzeti u obzir da je ta mogućnost specifična za natjecanje u STEM području i da veći broj ostalih studentskih natjecanja ne uključuje takav program.



Slika 2.1: Prikaz organizacijskog ustroja

2.3. Zadaci voditelja organizacije

Voditelji organizacije, za koje su predviđene dvije pozicije u organizacijskoj strukturi, započinju organizaciju projekta početkom akademske godine. Zaduženi su, najprije, za okupljanje predstavnika studenata institucija koje će sudjelovati.

Predstavници onih institucija koje su dosad sudjelovale na natjecanju čine tzv. *nadzorni odbor* koji potvrđuje voditelje organizacije za tu godinu. Voditelji organizacije potom digitalnim putem objavljuju natječaj za volontere u organizaciji, od kojih se biraju drugi članovi organizacije uz potvrdu nadzornog odbora.

Nakon ustroja organizacijskog tima, voditelji organizacije zaduženi su za vremensko i resursno planiranje organizacije. Moraju voditi računa o potencijalnim izvorima financiranja i rashodima potrebnima za organizaciju, konzultirati nadzorni odbor o željama sudionika te ih uzeti u obzir prilikom planiranja zadataka za organizacijske timove. Također, brinu se o suradnji s kooperantom koji omogućuje prostor u kojem će se odvijati natjecanje te posreduju u komunikaciji između ugostitelja i predstavnika institucija sudionika.

Osim toga, voditelji organizacije brinu se o neprofitnoj organizaciji koja stoji iza organizacijskog tima natjecanja. Brinu se o ispravnom bilježenju prihoda i rashoda u suradnji s računovodstvenim servisom, ugovorima sa sponzorima i kooperantima, održavanju skupština udruge i općenitim administrativnim poslovima koji su za nju vezani.

2.4. Zadaci studentskih predstavnika

Studentski predstavnici posreduju u komunikaciji između organizacijskog tima i studenata visokoobrazovnih institucija koje sudjeluju na natjecanju. Prema prethodnom sudjelovanju, neki od predstavnika studenata imaju pravo glasati o pitanjima vezanim uz tijek organizacije. To čine digitalnim putem ili na sastancima nadzornog odbora koje sazivaju voditelji organizacije.

Predstavници također biraju studente koje će njihova institucija povesti na natjecanje nakon što se studenti prijave preko jedinstvene stranice organizacije. Zaduženi su i za organizaciju prijevoza sudionika na događaj te eventualnog sufinanciranja njihovog sudjelovanja fakultetskim ili sponzorskim subvencijama.

2.5. Zadaci operativnog tima

Operativni tim kao organizacijski tim u užem smislu ima veliki broj zadataka, te je možda više od bilo kojeg drugog tima podložan informatizaciji. Glavne zadatke operativnog tima su briga o nabavi, upravljanje inventarom, organizacija lokalnog prijevoza,

upravljanje timom "junior" volontera², organizacija dodatnog programa, akreditacija sudionika te briga o rasporedu događanja.

Detaljni zahtjevi na informacijski sustav od strane ovog tima, kao i drugih timova, bit će opisani u nastavku, no ovdje vrijedi nešto detaljnije opisati pojedinosti nabrojanih zadaća.

2.5.1. Nabava i upravljanje inventarom

Operativni tim vodi evidenciju o svim kooperantima s kojima je ostvarena komunikacija tijekom tekućeg i prijašnjih projekata te prema tome ostvaruje suradnje nužne za provedbu događaja. Ostali timovi iskazuju svoje potrebe za nabavom materijala operativnom timu te članovi operativnog tima komuniciraju s potencijalnim dobavljačima i dogovaraju nabavu. Nabavu u skladu s definiranim budžetom odobravaju voditelji organizacije.

Veliki dio nabave potrebne za projekt odnosi se na tiskane materijale, koji se tijekom događaja koriste za promociju sponzora i informiranje sudionika. Prilikom nabave tiskanih materijala operativni tim mora, osim s dobavljačima, komunicirati s *timom za partnerstva* koji se brine o odobrenju materijala u kojima se spominju sponzori događaja te *medijskim i dizajn timom* koji priprema grafičke pripreme.

Tijekom događaja, tim upravlja svim inventarom koji se koristi za provedbu natjecanja. Brine o vlasništvu inventarskih stavki (one mogu biti posuđene, u vlasništvu udruge, volontera ili ugostitelja), zaduženjima i ovlastima među volonterima te lokacijama na kojima se pojedini predmeti u svakom trenutku nalaze. Također brinu o prijevozu tih predmeta na lokaciju i njihovom povratku vlasnicima.

2.5.2. Lokalni prijevoz i prijevoz organizacije

Jedan od zadataka operativnog tima također je osigurati prijevoz članova organizacijskog tima na mjesto događaja. Pritom mora voditi računa o lokaciji polaska različitih članova organizacije i rasporedu njihovog dolaska po danima.

Na lokaciji natjecanja mora brinuti o zaduženjima službenih vozila u najmu i prioritetima za njihovo korištenje. Također, mora koordinirati autobusni prijevoz sudionika u različitim komponentama natjecanja do sportskih dvorana ili dvorana u kojima se odvija natjecanje u znanju.

²volontera koji ne sudjeluju u organizaciji, ali pomažu fizički u provedbi samog događaja

2.5.3. Upravljanje sudionicima

Operativni tim zadužen je također za provedbu prijave sudionika preko web sučelja na stranicama organizacije, komunicirajući pritom s predstavnicima studenata o njihovim procedurama selekcije i kategorijama za koje žele otvoriti prijave. Nakon selekcija provedenih od strane matičnih institucija sudionika, operativni tim brine o njihovim osobnim podacima, izdaje ulaznice za događaj, brine o njihovom dolasku i odlasku te posreduje u komunikaciji s ugostiteljskom tvrtkom.

2.5.4. Dodatni program, raspored i dodatni volonteri

Uz glavni program natjecanja, na događanju STEM Games organizira se i dodatni program u obliku neslužbenih natjecanja, predavanja i slično. Koordinacija tog programa u smislu rasporeda, materijalnih i ljudskih resursa također je zadatak operativnog tima.

Na samom događaju, osim članova organizacijskog tima, organizaciji pomažu i volonteri u smislu fizičke i logističke potpore. Operativni tim zadužen je i za vodstvo spomenutog tima volontera od kojih je svaki dužan utrošiti unaprijed određeni broj sati dnevno na dodijeljena zaduženja. Jedna od zadaća volonterskog tima je i prihvati sudionika na dolasku, pri čemu je brzina obrade pristiglog sudionika vrlo važna i u čemu im informacijski sustav može pomoći.

2.6. Zadaci tima za partnerstva

Glavna zadaća partnerskog tima je omogućiti organizaciju događaja prikupljanjem sredstava od sponzora u industriji. Zadaće ovog tima posebno su podložne informatizaciji jer je za njihovo uspješno obavljanje potrebno pomno bilježiti svu dosadašnju komunikaciju vezanu za ostvarene i potencijalne suradnje.

Organizacija sponzorima nudi uslugu promocije kroz izlaganje i distribuciju promotivnih materijala te uključivanje u osmišljavanje problemskog dijela natjecanja. Te su usluge definirane kroz sponzorske pakete, koji ujedno predstavljaju i moguće kategorije sponzora.

Rezultati rada partnerskog tima u obliku ugovora o suradnji sa sponzorima usko su vezani uz financijske elemente organizacije te bi se stoga ta veza trebala moći pratiti u potencijalnom dijelu informacijskog sustava vezanom uz financijsko planiranje.

2.7. Zadaci tima za marketing i komunikacije te podtimova

Tim za marketing i komunikacije zadužen je za pružanje informacija o događaju široj javnosti te njegovu promociju. Pritom se koristi web stranicom, profilima na društvenim mrežama te medijskim partnerstvima. Svojim radom pruža vidljivost kako samom događaju, tako i sponzorskim institucijama koje omogućuju financijska sredstva za provedbu događaja. Pod taj tim također spadaju i timovi za grafički dizajn i fotografiju koji se brinu za kreiranje materijala potrebnih za promociju događaja.

S aspekta informatizacije, potrebe ovog tima svode se na praćenje ostvarenih i potencijalnih kontakata s medijskim partnerima, katalogizaciju fotografskih i video materijala te planiranje objava na kanalima digitalne komunikacije.

2.8. Timovi za natjecanja u sportu, znanju i računalnim igrama

Timovi za natjecanja bave se organizacijom sportskih i problemskih natjecanja te natjecanja u računalnim igrama. Njihovi zadaci tiču se sastavljanja rasporeda, zadataka za natjecatelje ili pripreme sportskih terena. Veći dio zadataka tih timova nije relevantan za razvoj informacijskog sustava, pa time ne spada u okvire ovog rada.

2.9. Ostale potrebe

Važan aspekt organizacijske strukture, a koji je zajednički svim timovima tiče se upravljanja pravima pristupa. Iako se ne radi o konkretnom organizacijskom zadatku, u svrhu potpunosti ovog opisa kao podloge za izolaciju zahtjeva, valja navesti problematiku.

Navedeni se timovi u svom radu koriste raznim internetskim servisima. Neki od tih servisa omogućuju individualnu registraciju korisnika, no kod nekih se korisnički podaci za prijavu dijele među više osoba.

Posljedica te prakse je da se osjetljivi podaci dijele u kanalima koji za to nisu predviđeni što uzrokuje sigurnosne rizike i poteškoće u praćenju korisnika koji u nekom trenutku imaju pristup usluzi.

Informacijski bi sustav u ovom aspektu mogao omogućiti sigurno dijeljenje pristupnih podataka prema potrebama i ovlastima članova organizacijskog tima. U slučaju

promjene osoba u ulogama ovlaštenim za pristup nekom servisu, zaporke bi se mogle jednostavnije promijeniti jer bi sustav omogućio da se o toj promjeni jednostavno obavijeste svi članovi tima koji koriste određeni servis.

Drugi aspekt kontrole pristupa je onaj fizički. Na samom događaju, uspostavljaju se organizacijski ured i skladište kojemu mogu pristupiti samo određeni članovi organizacije. Informatizacijom ovog aspekta pristup prostorima mogao bi se kontrolirati digitalnim putem, korištenjem uređaja već ustaljenih za tu namjenu poput RFID sustava, što bi otvorilo prostor za povećanu sigurnost u odnosu na korištenje tradicionalnih metoda poput korištenja papirnatih akreditacija.

3. Podaci organizacije za pohranu u IS

Nastavno na prethodni opis organizacije, a u svrhu ocjene prikladnosti ERP sustava otvorenog koda i kasnijeg razvoja prilagođenog rješenja, valja definirati potrebe organizacije koje bi informacijski sustav u primjeni trebao zadovoljavati.

S obzirom na prirodu organizacijskih potreba, prioritetni zahtjevi na informacijski sustav odnose se na pohranu podataka u sistematiziranom obliku te se kroz rad organizacije nije pokazala posebna potreba za njihovom obradom. Slijedom toga, u aplikaciji se ne zahtjeva sloj poslovne logike, te će se funkcionalni zahtjevi proučavati s aspekta podataka koje aplikacija mora pohranjivati. Implicitno će se podrazumijevati da aplikacija kroz svoje sučelje mora ponuditi operacije upisa, čitanja, modifikacije te brisanja (engl. *CRUD*), te da se prema pravilima organizacijske strukture za svaki od definiranih podatkovnih entiteta može upravljati ovlastima kategorija korisnika kojima je te operacije dopušteno izvoditi.

U ovom će poglavlju u svrhu analize zahtjeva opisno biti prikazane kategorije podataka koje organizacija mora moći pohraniti u sustavu te njihovi međusobni odnosi. Kako se od gotovih rješenja ne može očekivati podudarnost s potrebama organizacije u idealnom slučaju, prije njihove ocjene njihove prikladnosti neće se ulaziti u konkretne strukture podatkovnog modela, već će zahtjevi biti razmotreni na višoj razini. Razrada i formalizacija konkretnog modela relacijske baze podataka odgodit će se za poglavlje o rješenju korištenjem RAD programskog okvira.

3.1. Podaci o osobama

Organizacija natjecanja ovog tipa zahtjeva pohranu osobnih podataka velikog broja različitih kategorija osoba povezanih s događajem. Imajući u vidu osjetljivu prirodu osobnih podataka, vrlo je korisno omogućiti njihovu pohranu na jednom mjestu, kako s aspekta informacijske sigurnosti tako i zbog jednostavnosti upravljanja skupom podataka.

Osobni podaci koji se pohranjuju mogu pripadati članovima organizacijskog tima,

sudionicima natjecanja ili članovima drugih organizacija s kojima se surađuje tijekom organizacije natjecanja. Ovisno o kategoriji pojedinca čiji se podaci pohranjuju, tih podataka može biti više ili manje. Osnovni podaci uvijek su ime i prezime osobe, kontakt podatak koji može biti adresa elektroničke pošte ili broj telefona te pripadnost organizacijama i institucijama. Ovisno o potrebi, tome se može nadodati datum rođenja, adresa stanovanja, neka vrsta jedinstvenog identifikacijskog broja te u konkretnom slučaju opisane organizacije i veličina majice, u svrhu izrade promotivnih materijala.

Uz osobne podatke, moraju se pratiti i podaci o sudjelovanju pojedinca na konkretnom događaju. Ti će podaci biti opisani kasnije, ovisno o kategoriji sudjelovanja osobe u pitanju.

3.2. Korisnici sustava i uloge

Neke od osoba čiji bi se podaci bilježili u sustavu istovremeno bi bili korisnici tog sustava. U sustavu se moraju moći definirati korisnici uz svoje pristupne podatke te se različite korisnike mora moći povezati sa zapisima o njihovim osobnim podacima.

Svaki korisnik mora imati definirana prava pristupa različitim podacima pohranjenim u sustavu te prava njihove izmjene u smislu operacija upisa, modifikacije i brisanja. Kako se prava ne bi definirala pojedinačno za svakog korisnika, u sustavu mora postojati koncept uloge kao skupa prava koji se može unaprijed definirati i dodijeliti korisniku ovisno o njegovim organizacijskim potrebama i ovlastima.

Svaki korisnik može imati više uloga te svaka uloga može biti pridružena većem broju korisnika, a prava izmjene i dodjele uloga također su jedna od ovlasti koja može biti omogućena jednoj od uloga. Ovakav pristup sigurnosti, poznat kao RBAC (engl. *role-based access control*) vrlo je raširen (Ferraiolo D. (1992)), te je stoga izgledno da gotova rješenja nude takve mogućnosti.

3.3. Podaci o organizacijama i institucijama

Osim podataka o osobama uključenim u organizaciju događaja, važno je pohraniti podatke o raznovrsnim organizacijama i institucijama koje su važne za njegovu provedbu. To uključuje obrazovne institucije, sponzore, medijske partnere, kooperante te partnerske organizacije.

Većina podataka o organizaciji ovisi o prirodi angažmana te organizacije unutar projekta, te će stoga biti pobrojani u kasnijim potpoglavljima. Jedini obavezni podatak

u zapisu o pojedinoj organizaciji je njezino ime, a dodatni (fakultativni) podaci uključuju adresu sjedišta organizacije, njezin matični broj, relaciju prema kontakt osobama, generički kontakt organizacije te podatke o njezinom angažmanu na pojedinim projektima, poželjno u obliku relacije prema odvojenom zapisu o njezinom sudjelovanju.

3.4. Podaci o projektima

Svaki projekt, odnosno događanje koje se organizira, mora se moći zabilježiti u informacijskom sustavu uz svoje predviđene datume početka i završetka. U zapisu projekta nema potrebe za navođenjem ostalih podataka, ali se putem relacijskih veza moraju moći dohvatiti različiti zapisi koji se odnose na pojedini projekt.

Ti će zapisi biti nabrojani u nastavku, a uključuju dokumente, budžet, ugovore i račune, sponzorstva, sudjelovanja osoba, kooperante i vezane podatke.

3.5. Podaci o sudjelovanju obrazovne institucije

Osnovni podaci svake obrazovne institucije koja sudjeluje na natjecanju moraju biti zapisani u dijelu informacijskog sustava koji pohranjuje podatke o organizacijama i institucijama, no kao odvojenu kategoriju treba pohraniti i podatke o njihovom sudjelovanju na konkretnom projektu. Ti podaci uključuju relaciju prema projektu na kojem sudjeluju, relaciju prema osobama koje ispunjavaju ulogu predstavnika studenata za tu godinu, procjenu broja sudionika, broj glasova u nadzornom odboru na koji ta institucija eventualno ima pravo, te kategorije natjecanja u kojima želi sudjelovati.

Poželjno je da zapis o sudjelovanju obrazovne institucije bude izravno povezan sa zapisima o sudjelovanju njenih studenata na konkretnom natjecanju na koje se zapis o sudjelovanju odnosi.

3.6. Podaci o inventaru

Informacijski sustav mora omogućiti pohranu podataka o inventaru dostupnom organizaciji, koji može biti u vlasništvu organizacije, privatne osobe ili pravne treće osobe. Zapis o inventarskoj stavci, dakle, mora sadržavati ime te stavke po kojem ju se može identificirati, jedinstveni numerički identifikator, oznaku lokacije relaciju prema zapisu o vlasniku (osobi ili organizaciji) te relaciju prema osobi trenutno zaduženoj za brigu o stavci.

Dodatno, treba postojati mogućnost otpisa stavke, što može provesti brisanjem stavke ili označavanjem polja koje ukazuje na status otpisa. Koristeći tu mogućnost, zapisi o inventaru mogu se primijeniti i za praćenje potrošne robe.

3.7. Podaci o dokumentu

Uz fizički inventar organizacije, važno je pratiti i dokumente koji su od organizacijskog značaja. To mogu biti financijski, pravni ili organizacijski dokumenti, a glavni atributi zapisa o njima su naslov dokumenta, autor, vrijeme nastanka te lokacija (fizička ili virtualna) na kojoj se dokument nalazi.

Uz to, u svrhu lakšeg pretraživanja dokumenata, bilo bi korisno omogućiti upis kratkog opisa dokumenta.

3.8. Podaci o sponzoru

Prilikom komunikacije sa sponzorskim organizatorima, bilježenje povijesti kontakata može biti od velike pomoći partnerskom timu koji osigurava sredstva za održavanje događaja. Stoga je u informacijskom sustavu za potporu organizacijskom timu, potrebno omogućiti pohranu podataka o suradnji s pojedinom sponzorskom organizacijom.

Zapis o sponzorstvu poželjno bi relacijski bio vezan uz pojedini projekt i organizaciju, a sadržavao bi razinu interesa koju je sponzor iskazao, kratki opis sponzora, komentare o korespondenciji te relacijsku veze prema volonteru koji je odgovoran za komunikaciju sa sponzorom i eventualnim sklopljenim ugovorima.

Takav zapis omogućio bi detaljni uvid u povijest suradnje s pojedinim sponzorom u bilo kojem trenutku.

3.9. Podaci o sudjelovanju natjecatelja

Uz prijavu sudionika natjecatelja, za koju je važno imati zapis u informacijskom sustavu, osim osobnih podataka vežu se i podaci vezani uz sudjelovanje te osobe na pojedinom natjecanju. Ti podaci odnose se na kategorije u kojima se sudionik natječe, identifikacijsku oznaku tima za one discipline u kojima se natječu timovi te fakultativno smještajnu jedinicu u kojoj je sudionik smješten.

3.10. Podaci o sudjelovanju volontera

Važna organizacijska informacija jest i interni ustroj organizacije. S obzirom na praksu da na različitim projektima ista osoba može obavljati različite uloge, praktična funkcionalnost sustava bila bi da se za osobu koja radi na konkretnom projektu može odrediti njezina funkcija kroz pripadnost timu.

Zbog različitih zadataka koje članovi istog tima mogu obavljati, ovaj je koncept odvojen od uloge u smislu ovlasti korisnika sustava. Bilo bi, dakle, poželjno da sustav omogućuje pohranu zapisa o pripadnosti pojedine osobe projektним timovima te povezivanje tog zapisa sa zapisom o izravno nadređenom volonteru ako takav postoji.

3.11. Podaci o računima

Iako se o knjigovodstvu organizacije brinu za to stručni kooperanti, zbog interne evidencije sustav treba omogućiti osnovno praćenje izlaznih i ulaznih faktura. Izlazne fakture obično su vezane uz sponzore, a ulazne fakture uz kooperante. Uz to, svaka je faktura dokument, te bi zapisi o fakturama morali biti vezani uz pripadajuće zapise o dokumentima.

Svaki zapis o fakturi trebao bi sadržavati fakturirani iznos, datum izdavanja, datum dospijeca i podatak o tome je li plaćanje izvršeno. Dodatno, zapisi o izlaznim fakturama trebali bi sadržavati broj pod kojim je faktura izdana.

3.12. Podaci o vozilima

Tijekom provedbe samog natjecanja, organizacija mora voditi evidenciju o korištenju vozila koje uzima u najam. Stoga bi bilo korisno omogućiti praćenje planiranih i obavljenih vožnji vezanih uz pojedino vozilo.

Sustav bi u tu svrhu morao omogućiti upisivanje zapisa o vozilu te povezivanje tog zapisa sa zapisima o rezervacijama vozila za planirane vožnje kao i zapisima o obavljenim vožnjama. Svaki zapis o vožnji morao bi biti vezan uz osobu vozača, polazište i odredište te vrijeme polaska i povratka.

Koristan dodatak zapisima o vožnjama bio bi podatak o stanju spremnika za gorivo kako bi se za to zaduženim osobama omogućilo da na vrijeme opskrbe vozilo gorivom.

3.13. Podaci o budžetu

Uz svaki projekt, veže se i plan raspodjele očekivanih sredstava. Sustav mora omogućiti rudimentarno financijsko planiranje u obliku zapisa o kategoriji troška te njegovom iznosu. Poželjna nadogradnja bila bi da se ulazne fakture mogu povezati s nekim od zapisa u spomenutom planu, odnosno pojedinom budžetskom stavkom.

3.14. Podaci o kooperantima

Na isti način kako partnerski tim bilježi podatke o sponzorima, operativni tim trebao bi moći zabilježiti podatke o kooperantima. Zapis o kooperantu uključivao bi komentare o prethodnoj suradnji, relacijsku vezu s relevantnim zapisom o organizaciji te projektom na kojem se odvija suradnja te veze sa svim ulaznim fakturama izdanim od strane spomenutog kooperanta.

3.15. Pristupni podaci za mrežne usluge

Sustav bi također trebao omogućiti siguran prostor za pohranu pristupnih podataka vezanih uz korisničke račune na internetskim servisima koje koristi više osoba unutar organizacijskog tima. Svaki zapis o korisničkom računu mora sadržavati ime servisa kojem se pristupa, korisničko ime i zaporku, a relacijskom vezom mora biti ustanovljeno koje sve korisničke uloge imaju pravo pročitati pojedini zapis.

Sustav mora ograničiti pristup ovim podacima samo na one uloge koje su za korištenje tih podataka ovlaštene te lozinke mora pohranjivati u kriptografski zaštićenom formatu.

3.16. Podaci o medijskom partneru

Naposljetku, sustav mora omogućiti praćenje kontakata i korespondencije s medijskim partnerima događaja. Zahtjevi za ovu evidenciju identični su onima za evidenciju sponzorstava, no s obzirom na to da se odnosima s medijima bavi drugi tim, te dvije kategorije moraju biti razdvojene.

Dodatno, bilo bi poželjno omogućiti povezivanje medijskog partnera s medijskim objavama koje su mu predložene. Zapis medijske objave tada bi trebao povezivati zapis o dokumentu koji je izvorno izdao i poslao organizacijski tim, zapis o medijskom partnerstvu te, ukoliko postoji, poveznicu na objavu od strane medija.

4. Prikladnost *ERP* sustava otvorenog koda

Dio prethodno navedenih zadataka nije specifičan za organizaciju koja se bavi studentskim natjecanjima te se sasvim sigurno može riješiti korištenjem gotovog *ERP* rješenja. S obzirom na velike troškove uvođenja komercijalnog *ERP* rješenja, studentske se organizacije mogu služiti jedino *ERP* rješenjima otvorenog koda.

Fougatsaro (2009) u analizi devet takvih sustava zaključuje da, iako ih se pojavljuje sve više, velikim dijelom nisu zadovoljavajuće kvalitete — često su loše dokumentirani, nestabilni ili neodržavani. Iskustvo autora pri proučavanju *ERP* sustava koji uključuju module namijenjene neprofitnim organizacijama bilo je slično, te se kao jedina opcija vrijedna proučavanja nametnuo sustav naziva *ERPNext*

Sustav je dobro dokumentiran te za razliku od drugih rješenja otvorenog koda iza njega stoji komercijalno poduzeće koje uz naplatu nudi i korisničku podršku. Korisničko je sučelje intuitivno te se korištenjem *Docker* sustava *ERP* paket može pustiti u rad na jednostavan način. Kontrola pristupa pojedinim elementima sustava dobro je razrađena te omogućuje pripisivanje uloga ili grupa uloga pojedinim korisnicima, od kojih svaka može sadržavati prava pristupa pojedinom modulu. Sustav se također može integrirati s često korištenim poslovnim aplikacijama, od kojih neke koristi i organizacija opisana u ovom radu.

Očita prednost ovog sustava jest njegov modul za financije. Sustav omogućuje planiranje, praćenje troškova i prihoda te automatsko izdavanje računa koristeći obrasce zadane od strane korisnika. Mogućnosti ovoga sustava u financijskom aspektu sasvim sigurno nadilaze mogućnosti koje se mogu ostvariti koristeći *RAD* programski okvir.

Sustav kroz svoje module za neprofitne organizacije omogućuje i praćenje članova i volontera organizacije, no kako je namijenjen tradicionalnim neprofitnim organizacijama ne nudi funkcionalnost bilježenja hijerarhijske strukture timova prisutne u ovoj studentskoj organizaciji.

Kroz modul sustava namijenjen za upravljanjem klijentskim resursima, sustav omo-

gućuje i praćenje kontakata kooperanata i sponzora. Podatkovni model i korisničko sučelje u ovom su modulu relativno složeni te bi njegova primjena zahtijevala određenu edukaciju volontera korisnika, no to bi organizaciji moglo biti od koristi. Kroz isti sustav mogla bi se pratiti komunikacija s medijskim partnerima te bilježiti medijske objave.

Sustav također omogućuje praćenje inventara, no moduli za to daleko su složeniji od potreba studentske organizacije. Namijenjeni su za praćenje inventara velikih skladišta te uključuju pohranu brojnih nepotrebnih informacija (poput informacija o transportu, vrijednosti i amortizaciji, rokovima trajanja i slično). Primjena sustava u ovu svrhu uvela bi nepotrebne komplikacije te ne bi donijela veliku dodatnu vrijednost za organizaciju.

U sustavu, očekivano, nedostaju mogućnosti za praćenje sudjelovanja studenata na natjecanjima. Uz to, nedostaju mogućnosti praćenja vozila u posudbi te posuđenog inventara, sudjelovanja obrazovnih institucija i pohrane pristupnih podataka za internetske servise. Svaka od tih mogućnosti koje nedostaju mogla bi se implementirati kao novi modul u sustavu koristeći programski okvir dokumentiran u dokumentaciji sustava. Sustav se nadograđuje u programskom jeziku *Python* u kojem je i razvijen koristeći programski okvir *Frappe*.

5. Rješenje primjenom *RAD* programskog okvira

Kao drugi mogući pristup razmotrit će se informacijski sustav izrađen pomoću *RAD* (engl. *rapid application development*) programskog okvira. Takav programski okvir automatizira stvaranje obrazaca potrebnih za pohranu, pregled, uređivanje i brisanje podataka (engl. *CRUD*) na temelju definiranog relacijskog podatkovnog modela. Za razvoj informacijskog sustava koristeći takav programski okvir stoga je dovoljno definirati podatkovni model, željene prikaze, odnose među njima te eventualne nadogradnje u slučaju posebnih zahtjeva.

Pristup ovom problemu koristeći takav programski okvir prikladan je za studentsku organizaciju u pitanju utoliko što korisnicima s malo tehničkih znanja omogućuje jednostavan razvoj ili nadogradnju informacijskog sustava. Pristup tog tipa posljednjih se godina sve više uspješno primjenjuje te je poznat pod nazivom (engl. *low-code*) - s jedne strane pojednostavnjuje razvoj informacijskog sustava, no istovremeno ograničava fleksibilnost krajnje funkcionalnosti (Sahay et al. (2020)).

Cilj ovog dijela rada je ispitati praktičnost primjene jednog takvog alata razvijajući primjenjivu programsku podršku navedenim zahtjevima za slučaj *Udruge za studentska natjecanja STEM Games*. Razvijenom aplikacijom neće biti pokriveni svi navedeni zahtjevi, već oni koji su za rad Udruge po osobnom iskustvu autora važni te oni kroz koje se mogu ilustrirati primjene različitih mogućnosti koje nudi korišteni programski okvir. Time će se također nastojati dokumentirati proces oblikovanja informacijskog sustava kako bi se studentima volonterima olakšala njegova nadogradnja ili izmjena.

S tim ciljem, u nastavku ovog poglavlja najprije će biti razrađen relacijski podatkovni model koji će, uzevši u obzir broj entiteta i veza, iz praktičnih razloga biti prikazan i obrazložen kratkim opisom za svaku pripadajuću tablicu. Potom će na pojedinačnim primjerima biti opisan praktični aspekt implementacije pomoću programskog okvira *Flask-Appbuilder*, mogućnosti koje taj okvir pruža te problemi i ograničenja s kojima se autor susreo. Naposljetku, na nekoliko primjera bit će dokumentiran način

puštanja u rad te korištenja razvijene aplikacije.

5.1. Entiteti podatkovnog modela

U ovom potpoglavlju bit će opisani entiteti podatkovnog modela aplikacije razvijene korištenjem *RAD* programskog okvira. Praktični aspekt definicije podatkovnog modela korištenjem *Flask-Appbuilder* knjižnica bit će detaljnije razmotren u sljedećem poglavlju, no ovdje valja spomenuti da se isti određuje u objektnom obliku kroz kod u programskom jeziku *Python*.

Za svaki od entiteta definira se razred kojega knjižnica za objektno relacijsko preslikavanje (engl. *ORM*) potom prevodi u relacijske tablice. Stoga će u nastavku entiteti biti navedeni kao takvi, pod engleskim imenima razreda pod kojima su definirani u kodu. Za svaki od entiteta bit će navedeno kratko obrazloženje za njegovo uvođenje, opis njegovih atributa i veze prema drugim entitetima.

Veze između entiteta bit će navedene u opisu onog entiteta koji pohranjuje strani ključ druge strane veze. U slučaju relacija *više prema više* (engl. *many-to-many*), veza će biti navedena u opisu oba povezana entiteta.

5.1.1. Entitet *Project*

Ovaj se entitet odnosi na jedan projekt *Udruge*. Konkretna udruga za koju je razvijen model svake godine organizira jedno studentsko natjecanje koje je zasebni projekt te se brojni podaci koje sustav mora moći pohraniti povezuju s konkretnim projektom, što će biti navedeno u opisima odgovarajućih entiteta u nastavku.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime projekta

5.1.2. Entitet *Organization*

Ovaj entitet odnosi se na organizaciju bilo kojeg tipa s kojom je *Udruga* u nekom trenutku ostvarila suradnju ili planira ostvariti suradnju. Takva organizacija među ostalim može biti sponzor, kooperant ili obrazovna institucija, a eventualni detalji suradnje ili sudjelovanja pohranjeni su koristeći za to predviđene entitete koji su povezani s ovim.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime organizacije
- `address` – adresa organizacije (neobavezno)
- `generic_email` – kontakt elektroničke pošte organizacije (neobavezno)
- `oib` – OIB organizacije (neobavezno)

5.1.3. Entitet *Person*

Ovaj entitet pohranjuje podatke o svim osobama povezanim s *Udrugom*. Slično kao i s entitetom vezanim uz organizacije, takva osoba može biti sudionik, volonter, kontakt osoba druge organizacije ili bilo tko drugi povezan s radom *Udruge*, a za pohranu eventualnih detalja sudjelovanja koriste se za to predviđeni entiteti koji će biti opisani u nastavku.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `full_name` – ime i prezime
- `email` – adresa el. pošte (neobavezno)
- `phone` – kontakt broj telefona (neobavezno)
- `sex` – spol
- `oib_or_pin` – identifikacijski broj osobe (neobavezno)

5.1.4. Entitet *EduInstitution*

Ovaj entitet predstavlja obrazovnu instituciju. Povezuje se na entitet koji predstavlja organizaciju te je predviđen da sadrži dodatne detalje o obrazovnoj instituciji. U ovom modelu jedini je takav detalj kratica pod kojom se obrazovna institucija vodi u radu *Udruge*, no lako je zamisliti buduće nadogradnje po potrebi poput logotipa, područja studija i slično.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `abbreviation` – kratica imena
- `organization_id` – strani ključ pripadajuće organizacije

Ovaj entitet povezan je s entitetom *Organization* (potpoglavlje 5.1.2). Jedna obrazovna institucija obavezno je povezana s jednom organizacijom.

5.1.5. Entitet *OrganizationTeam*

Ovaj entitet predstavlja tim u organizacijskoj strukturi. S obzirom na to da je organizacijska struktura hijerarhijske naravi (vidi poglavlje 2.2), svaki tim može ili ne mora pripadati većem timu. Takva je stablasta struktura postignuta relacijom entiteta prema samome sebi.

Na ovaj entitet ne vežu se članovi timova, već se za svaki projekt definiraju timovi koristeći entitet *ProjectOrganizationTeam* (vidi 5.1.6). Ideja u pozadini ovog entiteta je da se pomoću njega definira organizacijska struktura koja je većim dijelom jednaka na svakom projektu te da služi kao predložak za definiranje timova na pojedinačnim projektima.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime tima
- `part_of_team` – ključ nadređenog, većeg tima (neobavezno)

Ovaj entitet ima povratnu vezu prema samome sebi.

5.1.6. Entitet *ProjectOrganizationTeam*

Ovaj entitet predstavlja organizacijski tim koji radi na jednom projektu. Praksa je *Udruga* da se za svaki projekt okuplja novi organizacijski tim pa se stoga pojavljuje potreba za praćenjem sudjelovanja volontera u različitim timovima za svaki projekt. Posljedično, koristeći ovaj entitet bilježi se sudjelovanje pojedinog volontera na pojedinom projektu.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `project_id` – strani ključ projekta kojem tim pripada
- `organization_team_id` – strani ključ tima u organizacijskoj strukturi

Ovaj tim povezan je s entitetom *OrganizationVolunteer* (vidi 5.1.7) koji predstavlja volontera u organizaciji u relaciji *više prema više* putem pomoćnih tablica *volunteer_team_membership* i *volunteer_team_leadership*. Prva od te dvije tablice povezuje tim s njegovim regularnim članovima, a druga s njegovim voditeljima. Jedan volonter može biti član ili voditelj više različitih timova te jedan tim može imati više od jednog voditelja i, naravno, člana.

5.1.7. Entitet *OrganizationVolunteer*

Ovaj entitet predstavlja volontera u organizaciji. Volonter može biti član ili voditelj jednog ili više projektnih timova (vidi potpoglavlja 5.1.6 i 5.1.5), te može biti aktivan ili neaktivan.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `person_id` – strani ključ zapisa o osobi

Entitet je povezan s entitetom *ProjectOrganizationTeam* kroz dvije relacije *više prema više* pomoću tablica spomenutih u potpoglavlju 5.1.6.

5.1.8. Entitet *InstitutionParticipation*

Ovaj entitet predstavlja sudjelovanje obrazovne institucije na pojedinom projektu. Istovremeno, uvodi vezu *više prema više* između obrazovnih institucija (vidi potpoglavlje 5.1.4) i projekata (vidi potpoglavlje 5.1.1), no smatrat će se entitetom zbog logike aplikacije te činjenice da također sadržava broj glasova koje pojedina institucija ostvaruje na projektu (vidi potpoglavlje 2.4). Ostaje također predviđena mogućnost da se ovaj entitet proširi drugim podacima o sudjelovanju institucije u slučaju da se za tim pokaže potreba.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `vote_number` – broj glasova institucije (0 ili više)
- `project_id` – strani ključ projekta
- `edu_institution_id` – strani ključ obrazovne institucije

Ovaj je entitet povezan s entitetima *Project* (potpoglavlje 5.1.1) i *EduInstitution* (potpoglavlje 5.1.4).

5.1.9. Entitet *InstitutionRepresentation*

Ovaj entitet predstavlja predstavnike studenata neke obrazovne institucije.¹ Svaka institucija može imati jednog ili više predstavnika, a predstavnik može biti glavni predstavnik ili zamjenik.

Atributi ovog entiteta su:

¹Funkcija predstavnika objašnjena je u potpoglavlju 2.4.

- `id` – primarni ključ
- `edu_institution_id` – strani ključ obrazovne institucije
- `person_id` – strani ključ zapisa o osobi
- `main_representative` – oznaka za predstavnike koji se smatraju "glavnima"

Ovaj je entitet povezan s entitetima *EduInstitution* (potpoglavlje 5.1.4) i *Person* (potpoglavlje 5.1.3).

5.1.10. Entitet *Location*

Ovaj entitet predstavlja fizičku lokaciju relevantnu za rad *Udruge*. U trenutnom modelu, zapisi o lokaciji služe kako bi se uz nju vezale inventarske stavke te fizički dokumenti, a predviđeno je da bi se na isti entitet mogli nadovezati podaci o pravima pristupa fizičkim lokacijama ukoliko se u budućnosti model proširi kako bi podržao i za to potrebne mehanizme.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime lokacije
- `address` – adresa lokacije (neobavezno)

5.1.11. Entitet *InventoryItem*

Ovaj entitet predstavlja inventarsku stavku i služi za praćenje imovine *Udruge* ili imovine trećih osoba i organizacija koju *Udruga* koristi u svom radu. Za svaku stavku bilježi se vlasnik (osoba ili organizacija), osoba koja je trenutno za nju odgovorna, te njezina lokacija. Također je predviđena mogućnost otpisa stavke u slučaju njezinog gubitka ili oštećenja te kako bi se isti entitet mogao koristiti za praćenje potrošnog materijala.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime stavke
- `owner_organization_id` – strani ključ organizacije vlasnice (neobavezno)
- `owner_person_id` – strani ključ osobe vlasnice (neobavezno)

- `responsible_person_id` – strani ključ odgovorne osobe
- `location_id` – strani ključ lokacije na kojoj se stavka nalazi
- `discarded` – oznaka otpisa

Ovaj entitet povezan je s entitetom *Person* (potpoglavlje 5.1.3) dvama vezama, jednom koja predstavlja vlasnika i jednom koja predstavlja osobu u čijem se zaduženju stavka trenutno nalazi. Također je povezan s entitetom *Organization* (potpoglavlje 5.1.2) relacijom koja se koristi u slučaju da je stavka u vlasništvu organizacije te s entitetom *Location* (potpoglavlje 5.1.2) kako bi se omogućilo bilježenje fizičke lokacije inventarske stavke.

5.1.12. Entitet *Sponsor*

Ovaj entitet predstavlja sponzorstvo neke organizacije na nekom pojedinom projektu, bilo ono ostvareno ili isključivo predloženo. Služi prvenstveno kako bi se sačuvala sva povijest komunikacije sa sponzorima na projektima te se na njega nadovezuju komentari vezani uz suradnju. Entitet također sadržava podatke o otvorenosti sponzora za suradnju na konkretnom projektu te kategoriju sponzorstva u slučaju da je suradnja ostvarena. Također sadrži podatak o osobi unutar organizacije koja je zadužena za komunikaciju s konkretnim sponzorom te se na njega vežu ugovori koji su s njime potpisani (vidi potpoglavlje 5.1.27).

Atributi ovog entiteta su:

- `id` – primarni ključ
- `organization_id` – strani ključ organizacije sponzora
- `responsible_person_id` – strani ključ osobe odgovorne za komunikaciju
- `project_id` – strani ključ projekta na koji se sponzorstvo odnosi
- `category_id` – strani ključ kategorije sponzorske suradnje
- `responsiveness_id` – strani ključ ocjene otvorenosti prema suradnji

Ovaj entitet povezan je s entitetima *Organization* (potpoglavlje 5.1.2), *Person* (potpoglavlje 5.1.3), *Project* (potpoglavlje 5.1.1), *SponsorCategory* (potpoglavlje 5.1.14), te *SponsorResponsivenessLevel* (potpoglavlje 5.1.15).

5.1.13. Entitet *SponsorComments*

Ovaj entitet predstavlja komentar bilo kakve vrste vezan uz suradnju s pojedinim sponzorom na pojedinom projektu.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `comment` – sadržaj komentara
- `sponsor_id` – strani ključ zapisa o sponzoru

Ovaj entitet povezan je s entitetom *Sponsor* (potpoglavlje 5.1.12).

5.1.14. Entitet *SponsorCategory*

Ovaj entitet (šifarnik) predstavlja moguće kategorije sponzorstva. Svaki sponzor s kojim je ostvarena suradnja svrstava se u jednu od kategorija sponzorstva.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime kategorije sponzorstva
- `description` – opis kategorije sponzorstva

5.1.15. Entitet *SponsorResponsivenessLevel*

Ovaj entitet (šifarnik) predstavlja otvorenost sponzora prema suradnji koju sponzor može pokazati na određenom projektu. Ocjene se, koristeći ovaj entitet, bilježe opisno te im se pridružuje brojčana vrijednost. U opis ocjene unose se kriteriji koje sponzor ocjenjen tom ocjenom zadovoljava.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime ocjene
- `description` – opis kriterija za ocjenu
- `score` – proizvoljna brojčana vrijednost pridružena ocjeni

5.1.16. Entitet *Subcontractor*

Ovaj entitet predstavlja podugovorenu organizaciju, odnosno kooperanta. Simetrično sa sponzorima (vidi potpoglavlje 5.1.12), *Udruga* prati podatke o suradnji ostvarenoj s pojedinim kooperantom na pojedinom projektu.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `project_id` – strani ključ projekta
- `organization_id` – strani ključ organizacije kooperanta

Ovaj se entitet, dakle, veže na entitete *Project* (potpoglavlje 5.1.1) i *Organization* (potpoglavlje 5.1.2).

5.1.17. Entitet *SubcontractorComments*

Ovaj entitet predstavlja komentar vezan uz suradnju s pojedinim kooperantom na pojedinom projektu. To mogu biti iskustva s uslugom, pristupačnost, općeniti zapisi o ostvarenim suradnjama i slično.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `comment` – sadržaj komentara
- `subcontractor_id` – strani ključ zapisa o kooperantu

Ovaj entitet povezan je s entitetom *Subcontractor* (potpoglavlje 5.1.16).

5.1.18. Entitet *PhysicalDocument*

Ovaj entitet predstavlja fizičku kopiju dokumenta od organizacijske važnosti. Takav dokument može biti ugovor ili račun (detaljnije razrađeno u ovom podatkovnom modelu, vidi potpoglavlja 5.1.27, 5.1.22 i 5.1.20) ili bilo kakav drugi dokument važan *Udrugi*. Entitet sadrži informacije o lokaciji kopije dokumenta, njegovom autoru, datumu izrade te kratki opis sadržaja.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime ili naslov dokumenta
- `location_id` – strani ključ lokacije dokumenta

- `author_id` – strani ključ autora dokumenta (neobavezno)
- `date` – datum izrade dokumenta
- `comment` – kratki opis dokumenta (neobavezno)

Ovaj entitet povezan je s entitetom *Person* (potpoglavlje 5.1.3) relacijom koja se odnosi na autorstvo te s entitetom *Location* (potpoglavlje 5.1.10).

Na ovaj entitet također se vežu prethodno spomenuti entiteti ugovora i računa koristeći pomoćne tablice što je detaljnije opisano u navedenim potpoglavljima vezanim uz te entitete.

5.1.19. Entitet *DigitalDocument*

Ovaj entitet, kao i prethodni, predstavlja kopiju organizacijski značajnog dokumenta, no u digitalnom obliku. Nalikuje entitetu *PhysicalDocument*, no umjesto navođenja fizičke lokacije kao referencu na dokument sadrži poveznicu na njegovu virtualnu lokaciju.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime ili naslov dokumenta
- `link` – poveznica na dokument
- `author_id` – strani ključ autora dokumenta (neobavezno)
- `date` – datum izrade dokumenta
- `comment` – kratki opis dokumenta (neobavezno)

Kao i prethodni, i ovaj entitet veže se na entitet *Person* (potpoglavlje 5.1.3) u svrhu zapisa autorstva dokumenta.

5.1.20. Entitet *InvoiceOutgoing*

Ovaj entitet predstavlja izlazni račun, odnosno račun izdan od strane *Udruge* prema sponzoru. Svaki izlazni račun sastoji se od jedne ili više stavki koje su predstavljene zasebnim entitetom (potpoglavlje 5.1.21), te se za njega bilježi tekstualna referenca, datumi izdavanja i dospijeca, organizacija platitelj i podatak o tome je li račun knjižen pri knjigovodstvenom servisu.

Iznos na koji se izdaje račun nije direktni atribut ovoga entiteta, već se on računa kao zbroj iznosa svih stavki povezanih s računom. Takav pristup odabran je kako bi se

mogle bilježiti odvojene stavke te kako bi se ilustrirao mehanizam izračunatih atributa kojega odabrani programski okvir nudi.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime računa (opisna referenca)
- `date_issued` – datum izdavanja računa
- `issue_to_id` – strani ključ organizacije kojoj je račun izdan
- `date_due` – datum dospeljeća računa (valuta plaćanja)
- `registered_with_accounting` – oznaka da je račun knjižen

Ovaj je entitet, dakle, povezan s entitetom *Organization* (potpoglavlje 5.1.2) kako bi se mogla zabilježiti organizacija kojoj je račun izdan.

Dodatno, pomoćnim tablicama *invoice_outgoing_digital_copies* i *invoice_outgoing_digital_copies* ovaj je entitet povezan s entitetima *PhysicalDocument* (5.1.18) i *DigitalDocument* (5.1.19). Iako je veza u stvarnosti tipa *jedan prema više*, ostvarena je pomoćnim tablicama kako bi struktura entiteta vezanih uz kopije dokumenata ostala nepromijenjena dodavanjem nove vrste dokumenta.

5.1.21. Entitet *InvoiceOutgoingItem*

Ovaj entitet predstavlja jednu stavku izlaznog računa. Stavka je jednostavni podatak koji se sastoji od imena i vrijednosti, te je povezana s točno jednim računom na kojem se nalazi.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime stavke
- `value` – iznos (cijena) stavke
- `invoice_outgoing_id` – strani ključ računa na kojemu se nalazi stavka

Ovaj entitet povezan je s entitetom *InvoiceOutgoing* (potpoglavlje 5.1.20).

5.1.22. Entitet *InvoiceIncoming*

Ovaj entitet predstavlja ulazni račun, odnosno račun kojega je dužna podmiriti *Udruga*. Osmišljen je imajući na umu broj takvih računa o kojima treba voditi evidenciju te je

stoga pojednostavljen tako da ne sadržava informacije koje za poslovanje *Udruge* nisu relevantne.

Podaci o računu koji se bilježe kroz ovaj entitet su tekstualna referenca, datum izdavanja, ime organizacije koja ga izdaje², iznos za plaćanje, datum dospijeća i oznaka je li račun knjižen pri knjigovodstvenom servisu. Ostavljen je također prostor za neobavezni komentar te broj računa pod kojim se račun vodi kod isporučitelja.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – tekstualna referenca računa
- `date_issued` – datum izdavanja računa
- `vendor_name` – ime organizacije isporučitelja
- `vendor_invoice_number` – broj računa (neobavezno)
- `value` – ukupni iznos za plaćanje
- `date_due` – datum dospijeća
- `comment` – tekstualni komentar (neobavezno)
- `registered_with_accounting` – oznaka da je račun knjižen

Kao i u slučaju entiteta vezanog uz izlazne račune (vidi potpoglavlje 5.1.20) i ovaj entitet vezan je uz svoje fizičke i digitalne kopije dvjema pomoćnim tablicama *invoice_outgoing_physical_copies* i *invoice_outgoing_digital_copies*.

5.1.23. Entitet *Income*

Ovaj entitet predstavlja prihod koji je *Udruga* ostvarila u nekom trenutku. Prihod može ili ne mora biti vezan uz neki izlazni račun, te ima iznos i datum ostvarivanja.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – tekstualni opis prihoda
- `value` – iznos prihoda
- `date_received` – datum izvršenja transakcije
- `invoice_incoming_id` – strani ključ ulaznog računa (neobavezno)

Ovaj je entitet, dakle, vezan za entitet *InvoiceIncoming* (potpoglavlje 5.1.22).

²Ovdje se iz praktičnih razloga račun ne povezuje s entitetom organizacije zbog velikog broja računa izdanih od strane organizacija za čijom evidencijom u sustavu nema potrebe.

5.1.24. Entitet *Expense*

Ovaj entitet predstavlja rashod (trošak) udruge. Slično kao prethodni entitet, za svaki rashod bilježi se opis, ime i datum transakcije, te se rashod može vezati uz ulazni račun. Dodatno, za rashod se bilježi i osoba koja ga je odobrila.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – tekstualni opis prihoda
- `value` – iznos prihoda
- `date_paid` – datum izvršenja transakcije
- `invoice_outgoing_id` – strani ključ izlaznog računa (neobavezno)
- `approved_by_id` – strani ključ osobe koja je odobrila rashod

Ovaj se entitet, dakle, veže uz entitete *InvoiceOutgoing* (potpoglavlje 5.1.20) te *Person* (potpoglavlje 5.1.3).

5.1.25. Entitet *PlannedIncome*

Ovaj entitet odnosi se na prihod koji je planiran. Sadrži opis prihoda, planirani iznos te planirani datum dospijeca prihoda. Ovaj entitet nije ni u kakvom odnosu s drugim entitetima modela.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – tekstualni opis
- `value` – novčani iznos
- `date_planned` – planirani datum transakcije

5.1.26. Entitet *PlannedExpense*

Ovaj se entitet odnosi na rashod koji je planiran. Njegova struktura identična je onoj prethodnog entiteta te također nije ni u kakvoj vezi s ostalim entitetima modela.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – tekstualni opis

- `value` – novčani iznos
- `date_planned` – planirani datum transakcije

5.1.27. Entitet *Contract*

Ovaj entitet odnosi se na ugovor koji je potpisan sa sponzorom na nekom određenom projektu. S obzirom na to da entitet koji predstavlja sponzora (potpoglavlje 5.1.12) već sadrži podatak o projektu na kojeg se taj sponzor odnosi, ovaj entitet ne bilježi direktno projekt na koji se odnosi. Podaci koje sadrži ovaj entitet su ime ugovora, potpisnik, sponzor s kojim se ugovor potpisuje, status (potpisan ili nepotpisan) te eventualna referenca na račun izdan prema ugovoru.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – tekstualna referenca (ime) ugovora
- `sponsor_id` – strani ključ sponzora potpisnika ugovora
- `internal_signee_id` – strani ključ osobe koja potpisuje ugovor unutar *Udruge*
- `signed` – status potpisa (potpisan ili nepotpisan)
- `invoice_outgoing_id` – strani ključ računa izdanog prema ugovoru (neobavezno)

Ovaj je entitet povezan s entitetima *Sponsor* (potpoglavlje 5.1.12), *Person* (potpoglavlje 5.1.3) te *InvoiceOutgoing*. Veza prema entitetu *Person* odnosi se na internog potpisnika ugovora.

Dodatno, entitet je povezan s entitetima *PhysicalDocument* (potpoglavlje 5.1.18) i *DigitalDocument* (potpoglavlje 5.1.19) pomoćnim tablicama *contract_physical_copies* i *contract_digital_copies* na isti način kako je to učinjeno za račune.³

5.1.28. Entitet *OrganizationContact*

Ovaj entitet predstavlja kontakt osobu neke organizacije. Povezuje organizacije i osobe, te sadrži podatak o ulozi osobe u organizaciji.

Atributi ovog entiteta su:

- `id` – primarni ključ

³Detalji ovakve veze objašnjeni su u potpoglavlju 5.1.20.

- `organization_id` – strani ključ organizacije
- `contact_person_id` – strani ključ osobe
- `role` – tekstualni opis uloge osobe u organizaciji

Entitet se veže na entitete *Organization* (potpoglavlje 5.1.2) i *Person* (potpoglavlje 5.1.3).

5.1.29. Entitet *ParticipantInfo*

Ovaj entitet predstavlja sudionika na jednom projektu. Povezuje se na entitet s osobnim podacima sudionika te njima pridodaje podatke vezane za sudjelovanje poput kategorije sudjelovanja, podatka o odobrenju sudjelovanja i veličine majice⁴. Zamišljeno je da se ovakav zapis vodi za svaku osobu koja sudjeluje na nekom događaju u organizaciji *Udruge*, a zapis se može povezati s organizacijom u čijoj odgovornosti sudionik sudjeluje ako takva postoji.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `person_id` – strani ključ osobe
- `project_id` – strani ključ projekta
- `responsible_organization_id` – strani ključ organizacije odgovorne za sudionika (neobavezno)
- `participation_approved` – oznaka odobrenja sudjelovanja
- `participation_approved_by_id` – strani ključ osobe koja je odobrila (ili treba odobriti) sudjelovanje
- `participation_approved` – oznaka odobrenja sudjelovanja
- `participation_category_id` – strani ključ kategorije sudjelovanja
- `t_shirt_size` – veličina majice (tekst)

Entitet je dvostruko povezan s entitetom *Person* (potpoglavlje 5.1.3) — jednom kako bi se zapis o sudioniku povezao s osobnim podacima sudionika, a drugi put putem relacije koja označava osobu odgovornu za odobrenje sudjelovanja konkretnog sudionika. Entitet je također vezan na entitete *Organization* (potpoglavlje 5.1.2) te *ParticipationCategory* (potpoglavlje 5.1.30).

⁴Veličina majice sudionika organizacijski je važan podatak zbog prakse tiska promotivnih majica.

5.1.30. Entitet *ParticipationCategory*

Ovaj entitet (šifarnik) definira moguće kategorije sudjelovanja sudionika. Takva kategorija može biti natjecateljska ili nenatjecateljska.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime kategorije

5.1.31. Entitet *Vehicle*

Ovaj entitet predstavlja vozilo kojim se koristi *Udruga*. Vozilo može biti u vlasništvu pojedinca ili organizacije (u slučaju najma ili posudbe). Entitet sadrži ime vozila (kratki opis), njegovu registarsku oznaku te oznaku mogućnosti rezervacije.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime vozila (kratki opis)
- `plate_number` – registarska oznaka vozila
- `available_to_reserve` – označava je li moguća rezervacija vozila
- `owner_person_id` – strani ključ fizičke osobe u čijem je vozilu vlasništvo (neobavezno)
- `owner_organization_id` – strani ključ organizacije u čijem je vozilu vlasništvo (neobavezno)

5.1.32. Entitet *VehicleReservation*

Ovaj entitet predstavlja rezervaciju vozila na događaju u organizaciji *Udruga*. Sadrži podatak o planiranom vremenu (početku i završetku rezervacije) te neobavezno podatak o stvarnim vremenima preuzimanja i povrata vozila.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `vehicle_id` – strani ključ vozila koje se rezervira
- `person_id` – strani ključ osobe koja rezervira
- `reservation_start_time` – datum i vrijeme početka rezervacije

- `reservation_end_time` – datum i vrijeme završetka rezervacije
- `vehicle_taken_time` – datum i vrijeme preuzimanja vozila (neobavezno)
- `vehicle_returned_time` – datum i vrijeme povrata vozila (neobavezno)

Entitet je povezan s entitetima *Vehicle* (potpoglavlje 5.1.31) i *Person* (potpoglavlje 5.1.3).

5.1.33. Entitet *ToolCredentials*

Ovaj entitet predstavlja pristupne podatke za internetske servise te se njime nastoji riješiti problem opisan u potpoglavlju 3.15. Entitet sadrži ime servisa na koji se podaci odnose, korisničko ime (ukoliko ono postoji) te zaporku koja se automatski pohranjuje u kriptiranom obliku.

Atributi ovog entiteta su:

- `id` – primarni ključ
- `name` – ime servisa na koji se odnose podaci
- `username` – pristupno korisničko ime (neobavezno)
- `password` – lozinka za pristup servisu (kriptirano)

Ovaj je entitet povezan s automatski generiranim entitetom *Role* vezom *više prema više* o čemu će biti riječi u nastavku.

5.1.34. Entiteti *User*, *Permission* i *Role*

Programski okvir koji je korišten za izradu ove aplikacije nudi automatske mehanizme za autorizaciju i kontrolu pristupa. Kao dio tih mehanizama, povrh korisnički definiranog modela sam programski okvir uvodi entitete *User*, *Permission* i *Role*.

Entitet *User* odnosi se na jednog korisnika sustava i sadrži osnovne podatke o korisniku te korisničko ime i *hash-vrijednost*⁵ potrebnu za autorizaciju korisnika. Programski okvir pritom nudi mogućnost proširivanja entiteta *User* što je u ovom konkretnom modelu iskorišteno kako bi se zapis o korisniku povezao sa zapisom o osobi u korisnički definiranom modelu.

Entitet *Permission* predstavlja dozvolu korisniku da obavi određenu radnju u sustavu. Programski okvir za svaki od definiranih entiteta automatski stvara odvojene dozvole za svaku od *CRUD* operacija kojima se potom može upravljati iz korisničkog

⁵vrijednost koju poprima kriptografska jednosmjerna funkcija na argumentu jednakom lozinki

sučelja aplikacije. Automatski se stvaraju i dozvole za prikaz pojedinih poveznica u izborniku korisničkog sučelja aplikacije te se njihovim dodjeljivanjem korisnicima može upravljati izgledom korisničkog sučelja koje će se pojedinom korisniku prikazati.

Entitet *Role* predstavlja uloge koje korisnici mogu poprimati, odnosno skupove dozvola definiranih kroz entitet *Permission* koji se mogu dodjeljivati korisnicima. Korisnik koji ima jednu ulogu poprima sve dozvole te uloge, te je taj entitet vezan prema entitetu *User* u odnosu *više prema više*.

Uz već spomenuto proširenje entiteta *User* vezom prema entitetu *Person*, autorizacijski sustav programskog okvira *Flask-Appbuilder* u ovom je slučaju dodatno proširen vezom *više prema više* prema entitetu *ToolCredentials*. Time se omogućava ograničavanje vidljivosti pojedinih pristupnih podataka na samo neke uloge u sustavu.

5.2. Implementacija pomoću *Flask-Appbuilder* programskog okvira

Prethodno opisani podatkovni model implementiran je u informacijskom sustavu koristeći programski okvir *Flask-Appbuilder*. Taj programski okvir nadogradnja je na programski okvir *Flask*, predviđen za razvoj web aplikacija u programskom jeziku *Python*. *Flask-Appbuilder*, kako je već spomenuto, automatizira veliki dio izrade aplikacije te je kompatibilan s različitim sustavima relacijskih i *NoSQL* baza podataka (Gaspar (2013)). Za rad s relacijskim bazama podataka koristi se knjižnica za objektno relacijsko preslikavanje (engl. *ORM*) *SQLAlchemy* koja se može povezati s većinom često korištenih relacijskih sustava baza podataka (engl. *RDBMS*) uključujući *SQLite*, *Postgresql*, *MySQL*, *Oracle* i *MS-SQL* (Bayer (2012)). Upotreba takve knjižnice omogućuje definiciju podatkovnog modela u *Python* kodu te potpunu neovisnost o relacijskom sustavu baze podataka u pozadini.

U nastavku će biti dokumentiran postupak implementacije informacijskog sustava koristeći spomenuti programski okvir na pojedinačnim primjerima. Postupak se sastoji od konfiguracije aplikacije, definicije podatkovnog modela te definicije prikaza korisničkog sučelja (engl. *ModelView*). Kroz pružene primjere nastojat će se sažeto prikazati sve potrebne radnje za proširenje ili izmjenu razvijene aplikacije prema budućim potrebama.

5.2.1. Programska okolina

Za razvoj aplikacije koristeći *Flask-Appbuilder* ili njezino puštanje u rad potrebno je osigurati odgovarajuću programsku okolinu. Preduvjet je, naravno *Python* prevoditelj (engl. *interpreter*) te programski paket *Flask-Appbuilder* sa svim paketima o kojima ovisi. Uz to, potrebna je neka vrsta relacijskog sustava baza podataka (engl. *RDBMS*) koji će pohranjivati podatke iz sustava te paket *cryptography* potreban za pohranu kriptiranih podataka (vidi potpoglavlje 5.1.33).

Za potrebe ovog rada korišten je *RDBMS SQLite* koji se isporučuje s *Python* prevoditeljem. U praktičnoj primjeni, bilo bi ga poželjno zamijeniti robusnijim sustavom što se može postići izmjenom konfiguracijske datoteke aplikacije spomenute u nastavku.

Programski paketi korišteni za razvoj aplikacije iz ovog rada mogu se instalirati na sustav na kojemu je prisutan *python* prevoditelj pokretanjem sljedeće naredbe:

```
> python -m pip install Flask-Appbuilder cryptography
```

koja će pokretanjem programa za upravljanje paketima instalirati dva navedena paketa i sve pakete o kojima ovise.⁶

Paket *Flask-Appbuilder* isporučuje se s pomoćnim alatom *fab* koji među ostalim omogućuje automatsko generiranje strukture datoteka izvornog koda aplikacije, kreiranje relacijske baze podataka prema definiciji modela te stvaranje početnog administratorskog korisnika (Gaspar (2013)). U slučaju izrade nove aplikacije, kostur izvornog koda može se stvoriti pokretanjem naredbe:

```
> flask fab create-app
```

čime će se pokrenuti interaktivni alat koji će stvoriti strukturu direktorija i datoteka predviđenih za izvorni kod aplikacije (Gaspar (2013)). Struktura paketa izvornog koda razvijenog tijekom izrade ovog rada prikazana je u dodatku A ovom radu.

U slučaju pokretanja već razvijene aplikacije, nije potrebno izvršiti navedenu naredbu. Kako bi se aplikacija pustila u rad, u konfiguracijskoj datoteci `config.py` (vidi dodatak A) potrebno je postaviti putanju prema željenoj bazi podataka:⁷

```
SQLALCHEMY_DATABASE_URI = <putanja prema bazi podataka>
```

Potom je potrebno izvršiti naredbu:

⁶Praksa je da se za instalaciju *Python* okruženja aplikacija koriste tzv. virtualne okoline (engl. *virtual environment*), no detalji tog pristupa izlaze iz opsega uputa u ovom radu.

⁷Putanja mora biti u formatu kojega prihvata *SQLAlchemy ORM*. Format putanje opisan je u dokumentaciji programskog paketa te komentarima koje *Flask-Appbuilder* automatski dodaje u konfiguracijsku datoteku.

```
> flask fab create-db
```

koja će automatski generirati bazu podataka prema modelu definiranom u datoteci izvornog koda `models.py` (vidi dodatak A) (Gaspar (2013)).

Naposljetku, valja dodati administratorskog korisnika korištenjem naredbe:

```
> flask fab create-admin
```

kojom se pokreće interaktivni alat kojim se može dodati novi administrator (korisnik sustava kojemu su dodijeljene sve postojeće dozvole) (Gaspar (2013)).

Nakon inicijalizacije, aplikacija je spremna za puštanje u rad. U razvojnoj okolini, to se može postići naredbom:

```
> flask run
```

koja će pokrenuti ugrađeni *HTTP* poslužitelj i početi posluživati aplikaciju.

Prilikom puštanja aplikacije u rad u stvarnom okruženju, poželjno je ipak ne koristiti poslužitelj ugrađen u *Flask* programski paket već neki od sigurnijih i robusnijih *HTTP* poslužitelja (Grinberg (2018)). To je omogućeno činjenicom da *Flask* aplikacije odgovaraju specifikaciji *WSGI* (engl. *Web Server Gateway Interface*). Ta specifikacija određuje sučelje za Web aplikacije razvijene u *Python* programskom jeziku koje im omogućuje povezivanje s velikim brojem poslužiteljskih programa (Eby (2010)) poput danas popularnih *Apache* i *Nginx*.

5.2.2. Definicija modela

U ovom potpoglavlju na nekoliko će primjera biti prikazan postupak definicije entiteta u *Python* kodu te će se kroz njih nastojati ilustrirati funkcionalnosti *SQLAlchemy* knjižnice korištene u razvoju aplikacije za potrebe ovog rada.

Podatkovni model definiran je u datoteci `models.py` (vidi dodatak A).

Listing 5.1: kod entiteta *Expense*

```
class Expense(Model):
    id = Column(Integer, primary_key=True)
    name = Column(String(200), nullable=False)
    invoice_incoming_id = Column(Integer, ForeignKey('invoice_outgoing.id'))
    invoice_incoming = relationship('InvoiceIncoming', backref=backref('expense'))
    value = Column(Numeric, nullable=False)
    approved_by_id = Column(Integer, ForeignKey('person.id'), nullable=False)
    approved_by = relationship('Person')
    date_paid = Column(Date, nullable=False)

    def __repr__(self):
        return self.name
```

Iz isječka koda 5.1 vidljiv je osnovni princip definicije entiteta. Entitet se oblikuje kao *Python* objekt izveden iz objekta *Model* kojega nudi paket *SQLAlchemy*, a njegovi se atributi definiraju kao atributi *Python* razreda. Svakom atributu pridružuje se objekt razreda *Column* koji definira pojedini atribut (Bayer (2012)).

Rezultat ovakvog pristupa je razred čiji se objekti ponašaju kao obični *Python* objekti, a automatski su povezani s relacijskom bazom podataka u pozadini. Ako nije izričito zadano drugačije, za svaki od takvih razreda u bazi podataka stvorit će se odgovarajuća tablica čije ime odgovara imenu razreda uz prilagodbu konvencije imenovanja (npr. *InvoiceIncoming* postat će *invoice_incoming*).

Na primjeru se može vidjeti i mehanizam ostvarivanja veza između entiteta. Strani se ključevi definiraju kao i ostali atributi razreda, odnosno entiteta, a potom se definiraju dodatni atributi razreda kojima se pridružuju veze korištenjem funkcije *relationship*. Pristupanjem tim dodatnim atributima unutar *Python* koda pristupa se direktno objektu koji je povezan stranim ključem.

Pri ostvarivanju takve veze, korištenjem funkcije *backref* može se ostvariti i povratna veza. U konkretnom primjeru, korištenjem te funkcije automatski se definira atribut *expense* na objektima razreda *InvoiceIncoming*. Pristupanjem tom atributu iz *Python* koda može se dohvatiti povezani objekt razreda *Expense*, a u slučaju veza tipa *više prema jedan* ili *više prema više* takvim postupkom dohvaća se lista svih povezanih objekata. Definicija ovakvih povratnih veza u ovom se slučaju direktno reflektira i na korisničko sučelje aplikacije gdje će se one automatski prikazati.

Listing 5.2: kod entiteta *InvoiceOutgoing*

```
class InvoiceOutgoing(Model):
    id = Column(Integer, primary_key=True)
    name = Column(String(200), nullable=False)
    date_issued = Column(Date, nullable=False)
    issued_to_id = Column(Integer, ForeignKey('organization.id'))
    issued_to = relationship('Organization')
    date_due = Column(Date, nullable=False)
    digital_copies = relationship('DigitalDocument',
                                secondary=invoice_outgoing_digital_copies)
    physical_copies = relationship('PhysicalDocument',
                                secondary=invoice_outgoing_physical_copies)
    registered_with_accounting = Column(Boolean)

    @hybrid_property
    def value(self):
        return sum(i.value for i in self.items)

    @value.expression
    def value(cls):
        return (
```

```
select ([ func.sum( InvoiceOutgoingItem . value ) ] ).
where( InvoiceIncomingItem . invoice_incoming_id == cls . id ).
label( ' value ' )
)
```

Isječak koda 5.2 dodatno ilustrira dvije korištene funkcionalnosti *SQLAlchemy* paketa.

Prva je ostvarivanje veza korištenjem sekundarnih tablica (primjer definicije takve tablice dan je u isječku koda 5.3), koje se navode kao argument prethodno spomenute funkcije *relationship*.

Druga je korištenje tzv. *hibridnih atributa* kojima se entitetu daje virtualni atribut izračunat iz podataka prisutnih u bazi. Takav se atribut ostvaruje dvjema definicijama metoda - u prvoj definiciji (označenoj dekoratorom *hybrid_property* definira se izraz prema kojemu se virtualni atribut računa kad mu se pristupa iz *Python* koda, a u drugoj se isti izraz definira kao *SQL* upit kako bi se njegov rezultat mogao koristiti u složenijim upitima na bazu. *ORM* paket na takvom entitetu omogućava slaganje upita na transparentan način, odnosno virtualni se atribut ponaša na potpuno isti način kao što bi se ponašao da se radi o stvarnom atributu entiteta.

Listing 5.3: kod pomoćne tablice *invoice_outgoing_digital_copies*

```
invoice_outgoing_digital_copies = Table('invoice_outgoing_digital_copies',
    Model.metadata,
    Column('id', Integer, primary_key=True),
    Column('invoice_outgoing_id', Integer, ForeignKey('invoice_outgoing.id')),
    Column('digital_document_id', Integer, ForeignKey('digital_document.id')),
    unique=True))
```

5.2.3. Definicija korisničkog sučelja

Nakon definicije modela, potrebno je definirati prikaze korisničkog sučelja koji će se prikazati korisnicima. Programski paket *Flask-Appbuilder* taj postupak krajnje pojednostavljuje.

Definicije prikaza korisničkog sučelja nalaze se u datoteci *views.py* (vidi dodatak A).

Listing 5.4: kod prikaza *SponsorModelView*

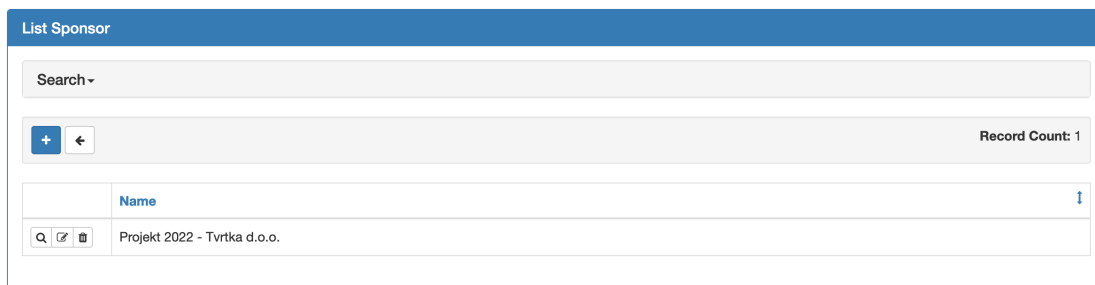
```
class SponsorModelView(ModelView):
    datamodel = SQLAlchemyInterface(Sponsor)
    related_views = [SponsorCommentsModelView]
    show_template = 'appbuilder/general/model/show_cascade.html'

    list_columns = ['name']
```

Na primjeru u isječku koda 5.4 može se vidjeti oblik jedne takve definicije. Za svaki prikaz definira se razred izveden iz razreda `ModelView`. Taj se razred s podatkovnim entitetom povezuje putem atributa `datamodel` što rezultira automatskim stvaranjem *master-detail* prikaza vezanog uz taj entitet u korisničkom sučelju koji nad njim omogućava *CRUD* operacije.

Prikazi se mogu međusobno povezivati koristeći atribut razreda `related_views`. Tom se atributu pridružuje lista drugih prikaza koji su u nekom odnosu s entitetom na koji se prikaz odnosi, nakon čega će sustav u *detail* prikazu automatski omogućiti baratanje povezanim zapisima. Ovdje valja napomenuti da povezani prikazi moraju biti definirani prije njihovog navođenja u spomenutom atributu.

Dodatno, unutar atributa `list_columns` potrebno je navesti sva podatkovna polja koja će se pojaviti u tabličnom *master* prikazu. Imena polja ovdje se odnose na attribute entiteta. Također je moguće odabrati način prikazivanja povezanih prikaza postavljanjem atributa `show_template`, pri čemu sustav nudi prikaz u karticama ili na istoj stranici.



Slika 5.1: *Master* prikaz vezan uz entitet `Sponsor`

Show Sponsor

Organization	Tvrtka d.o.o.
Responsible Person	Pero Perić
Project	Projekt 2022
Category	None
Responsiveness	None
Comments	[Primjer komentara 1]
Contracts	[]

←

List Sponsor Comments ▾

+

←

Record Count: 1

	Sponsor	Comment
<div> <div>Q</div> <div>✎</div> <div>✖</div> </div>	Projekt 2022 - Tvrtka d.o.o.	Primjer komentara 1

Slika 5.2: *Detail prikaz vezan uz entitet Sponsor*

Na kraju je potrebno registrirati definirani prikaz te odrediti njegov položaj u strukturi izbornika. To se postiže na način vidljiv u isječku koda 5.5.

Listing 5.5: registracija prikaza

```

appbuilder.add_view(
    SponsorModelView,
    "Sponsors",
    icon = "fa-folder-open-o",
    category = "Partnerships",
    category_icon = "fa-handshake-o"
)

```

5.2.4. Filtriranje i kontrola pristupa

Kao što je već spomenuto u potpoglavlju 5.1.34, na primjeru entiteta *ToolCredentials* proširen je sigurnosni sustav ugrađen u *Flask-Appbuilder*. Ugrađeni sigurnosni sustav omogućuje upravljanje pravima pristupa registriranim prikazima (vidi prethodno potpoglavlje) te pojedinačnim *CRUD* operacijama, no ne omogućuje kontrolu pristupa pojedinačnim zapisima u bazi podataka.

Stoga je u svrhu ograničavanja vidljivosti zapisa entiteta *ToolCredentials* potrebno uvesti dodatni mehanizam. U aplikaciji razvijenoj za potrebe ovog rada, to je ostvareno uvođenjem relacije *više prema više* između entiteta *ToolCredentials* i *Role* koristeći pomoćnu tablicu definiranu kao razred *ToolCredentialPermission*.

Za tu tablicu definiran je posebni prikaz na način opisan u prethodnom poglavlju koji omogućuje korisnicima koji za to imaju ovlasti da povezuju uloge u sustavu s podacima za pristup pojedinom servisu. Prikaz entiteta *ToolCredentials* potom je definiran tako da podatke prije prikaza filtrira prema dozvolama uloga koje su pripisane trenutnom korisniku.

To se može postići korištenjem atributa `base_filter` prilikom definiranja prikaza za taj entitet. Ova je mogućnost izrazito slabo dokumentirana u dokumentaciji *Flask-Appbuilder* paketa, no njezino je korištenje ilustrirano u programskim primjerima dostavljenim s tim paketom. Iz tog razloga, vrijedi izdvojiti i ukratko komentirati taj dio koda.

Listing 5.6: filtriranje pristupnih podataka

```
class CredentialsFilterFunction(BaseFilter):
    def apply(self, query, func):
        query, field = get_field_setup_query(query, self.model, self.column_name)
        return query.filter(field.any(ToolCredentials.id.in_(func())))

    def get_user_allowed_credentials():
        return set(c.id for role in g.user.roles \
                   for c in role.accessible_credentials)
    ...

class ToolCredentialsModelView(ModelView):
    datamodel = SQLAlchemyInterface(ToolCredentials)
    related_views = [ToolCredentialPermissionModelView]
    list_columns = ['tool_name', 'username', 'password']
    base_filters = [('have_access',
                    CredentialsFilterFunction, get_user_allowed_credentials)]
```

Najprije treba definirati razred izveden iz razreda `BaseFilter` definiranog u *Flask-Appbuilder* programskom paketu. U tom razredu treba definirati metodu `apply` koja prima *SQLAlchemy* upit kao prvi parametar i funkciju koja pri svojem pozivu vraća sve ključeve zapisa dostupnih korisniku. Ta metoda mora vratiti upit nadopunjen željenim filterom.

Potom treba definirati funkciju koja će vratiti ključeve zapisa za pristup trenutno prijavljenom korisniku, kojem se može pristupiti koristeći objekt `g` dostupan u knjižnici *Flask*. U ovom je to slučaju funkcija `get_user_allowed_credentials`.

Na kraju, u prikazu koji se filtrira treba definirati atribut razreda `base_filters` te u njemu navesti vezu po kojoj se filtrira, te definirani razred i funkciju. Konkretna definicija može se vidjeti u isječku koda 5.6.

Dodatno, sigurnosni mehanizam u ovoj je aplikaciji proširen i dodavanjem veze između entiteta *User* i *Person* (vidi potpoglavlje 5.1.34). Takvo proširenje postiže se

definiranjem novih razreda izvedenih iz razreda *User* i *UserModelView*, no taj postupak dobro je dokumentiran u dokumentaciji programskog paketa *Flask-Appbuilder* te stoga njegov opis izlazi iz opsega ovog rada.

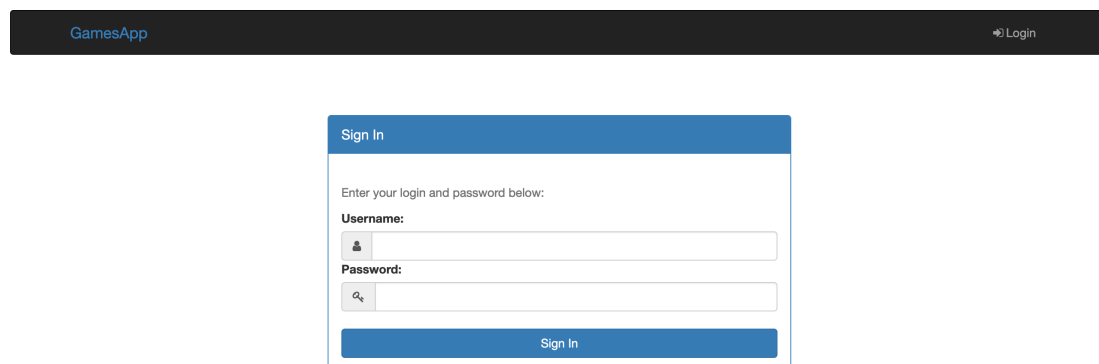
5.3. Korištenje aplikacije

U ovom potpoglavlju ukratko će biti opisan način korištenja razvijene aplikacije. S obzirom na veliki broj već opisanih entiteta te kako bi se izbjeglo ponavljanje, mogućnosti aplikacije bit će predstavljene na nekoliko primjera podatkovnih entiteta pri čemu će se podrazumijevati da se s ostalim entitetima postupa na jednak način.

5.3.1. Glavno sučelje

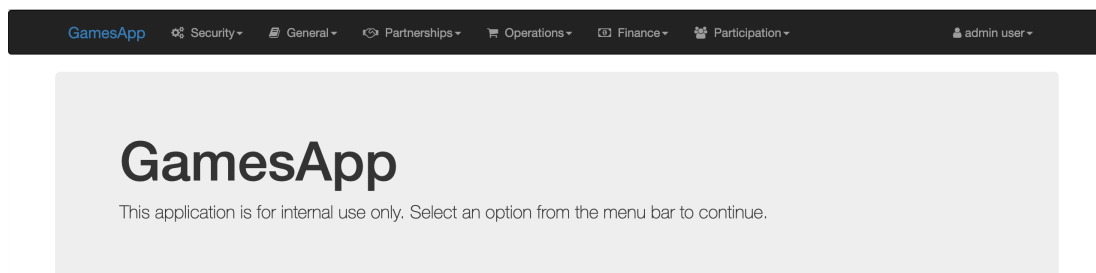
Otvaranjem početne stranice, neprijavljenom korisniku nudi se opcija za prijavu. Sučelje za prijavu korisnika *Flask-Appbuilder* generira automatski, a početni administratorski korisnik može se registrirati koristeći naredbe iz potpoglavlja 5.2.1.

Flask-Appbuilder nudi i automatsko sučelje za samostalnu registraciju korisnika, no u ovoj primjeni ta mogućnost nije korištena jer nije poželjno omogućiti registraciju neovlaštenih korisnika.



Slika 5.3: Sučelje za prijavu korisnika

Nakon prijave, korisniku se prikazuje početna stranica s alatnim izbornikom. Opcije dostupne u izborniku za pojedinog korisnika mogu se ograničiti izborom odgovarajuće uloge ili dozvola unutar uloge, a administratorskom korisniku prikazat će se sve dostupne opcije.



Slika 5.4: Početna stranica administratora

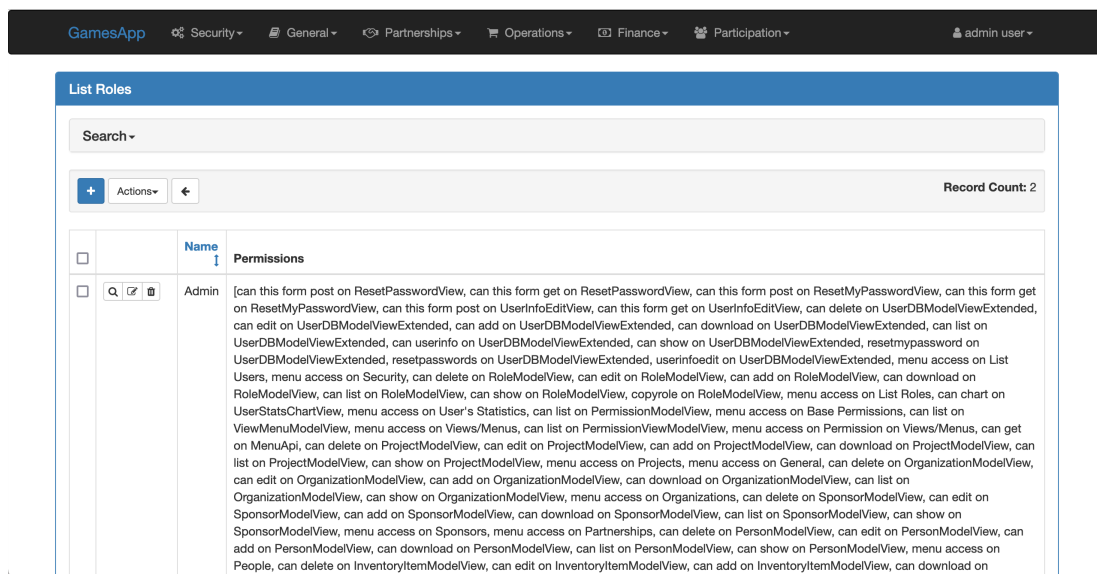
U alatnom izborniku dostupni su svi prikazi registrirani u datoteci `views.py` kojima je trenutnom korisniku dozvoljeno pristupiti. U ovoj aplikaciji, takav je prikaz definiran za svaki od entiteta opisanih u potpoglavlju 5.1.

Prikazi su podijeljeni u nekoliko kategorija, odnosno izbornika. U dodatku B rada mogu se vidjeti svi dostupni izbornici.

Dodatno, u alatnoj se traci pojavljuje i izbornik *Security* koji omogućuje upravljanje ugrađenim mehanizmima kontrole pristupa. U taj izbornik u razvijenoj aplikaciji dodan je i prikaz *Tool Credential Permissions* kojim se upravlja pravima pristupa zapisima u prikazu *Tool Credentials*.

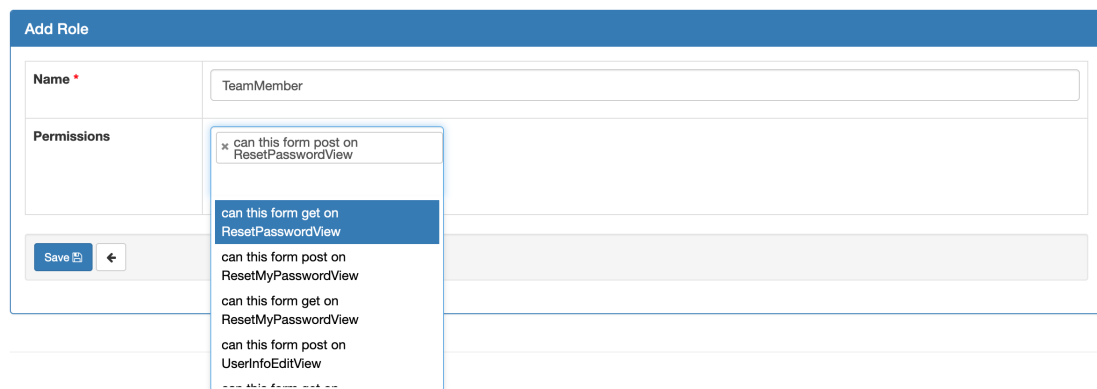
5.3.2. Upravljanje korisnicima, dozvolama i ulogama

Administrator nakon početka korištenja aplikacije mora definirati uloge korisnika u sustavu te dodati korisnike. Ovlasti koje se mogu pripisati određenoj ulozi unaprijed su prisutne u sustavu, a ulogama se može upravljati odabirom opcije *List Roles* iz izbornika *Security*.



Slika 5.5: Prikaz *List Roles*

Stvaranje nove uloge može se postići pritiskom na gumb označen znakom plus na otvorenom prikazu. Potrebno je odabrati ime uloge te odgovarajuće dozvole u sustavu koje toj ulozi pripadaju. Kao što se na slici 5.6 može vidjeti, sučelje za odabir dozvola u paketu *Flask-Appbuilder* izvedeno je na izrazito nepraktičan način.



Slika 5.6: Dodavanje uloge

Administrator ili za to ovlašten korisnik potom može pregledavati, uređivati ili dodavati korisnike odabirom opcije *List Users*, također iz izbornika *Security*. Prilikom dodavanja korisnika, administrator odabire njegove uloge, postavlja inicijalnu lozinku te eventualno povezuje korisnika sa zapisom o osobi u bazi podataka. Korisnik će potom moći promijeniti svoju lozinku nakon što se prijavi s onom inicijalno postavljenom.

Add User

First Name *	<input type="text" value="Pero"/> Write the user first name or names
Last Name *	<input type="text" value="Perić"/> Write the user last name
User Name *	<input type="text" value="pero"/> Username valid for authentication on DB or LDAP, unused for OID auth
Is Active?	<input checked="" type="checkbox"/> It's not a good policy to remove a user, just make it inactive
Email *	<input type="text" value="pero@peric.hr"/> The user's email, this will also be used for OID auth
Role	<div> × TeamMember </div> The user role on the application, this will associate with a list of permissions
Person	<div> </div>
Password *	<input type="password" value=""/> The user's password for authentication
Confirm Password *	<input type="password" value=""/> Please rewrite the user's password to confirm

Save

Slika 5.7: Dodavanje korisnika

Svaki korisnik može biti aktivan ili neaktivan. Sustav omogućuje i predlaže administratoru da korisnike deaktivira umjesto brisanja.

5.3.3. Upravljanje podacima

Na primjeru dodavanja jednog sponzorstva bit će ilustrirano postupanje s podacima u sustavu. Sučelje za upravljanje svim ostalim entitetima navedenim u poglavlju 5.1 funkcionira na identičan način.

Odabirom opcije *Sponsors* iz izbornika *Partnerships* otvara se popis sponzorstava.

List Sponsor

Search

+

Record Count: 1

Name
<div> <div> <div>Q</div> <div>✎</div> <div>🗑</div> </div> <div>Projekt 2022 - Tvrtka d.o.o.</div> </div>

Slika 5.8: Popis sponzorstava

Pritiskom na gumb označen znakom plus moguće je dodati novog sponzora.

Organization *	Tvrtka d.o.o.
Responsible Person *	Pero Perić
Project *	Veliki Projekt 2021
Category	Generalni Sponzor
Responsiveness	Otvoren za suradnju
Comments	Select Value
Contracts	Select Value

Save + ↩

Slika 5.9: Dodavanje sponzora

Dodani sponzor potom će se pojaviti u glavnom prikazu.

List Sponsor	
Search	
+ ↩ Record Count: 2	
	Name ↑
🔍 📄 🗑️	Projekt 2022 - Tvrtka d.o.o.
🔍 📄 🗑️	Veliki Projekt 2021 - Tvrtka d.o.o.

Slika 5.10: Sponzor dodan na popis

Podaci se potom mogu filtrirati dodavanjem filtra u okviru *Search*.

List Sponsor

Search ▾

Add Filter ▾

× Project

Relation ▾

Veliki Projekt 2021

Projekt 2022

Veliki Projekt 2021

Search Q

+ ←

Record Count: 2

	Name	1
Q []	Projekt 2022 - Tvrtka d.o.o.	
Q []	Veliki Projekt 2021 - Tvrtka d.o.o.	

Slika 5.11: Definiranje filtra

Pritiskom na gumb označen ikonom povećala pokraj zapisa u glavnom prikazu otvara se detaljni prikaz zapisa u kojem su dostupni i povezani prikazi (u ovom slučaju prikaz koji se odnosi na komentare o sponzoru).

Show Sponsor

Organization	Tvrtka d.o.o.
Responsible Person	Pero Perić
Project	Veliki Projekt 2021
Category	Generalni Sponzor
Responsiveness	Otvoren za suradnju
Comments	[Zainteresiran za promociju vlastite mobilne aplikacije]
Contracts	[]

←

List Sponsor Comments ▾

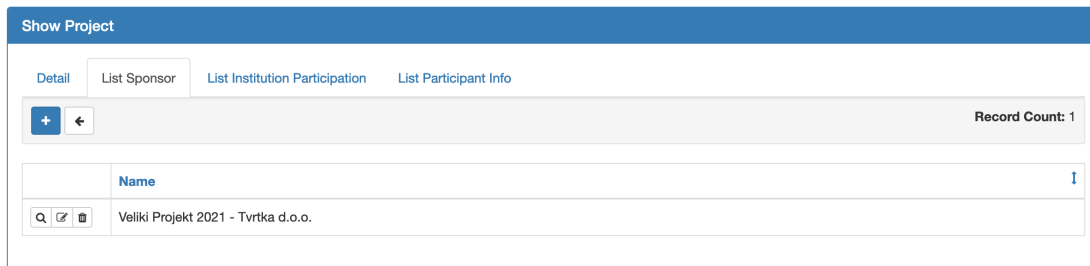
+ ←

Record Count: 1

	Sponsor	Comment	1
Q []	Veliki Projekt 2021 - Tvrtka d.o.o.	Zainteresiran za promociju vlastite mobilne aplikacije	

Slika 5.12: Detaljni prikaz zapisa o sponzoru

Podatak o sponzoru sada je također dostupan u detaljnom pregledu projekta pod odgovarajućom karticom.



Slika 5.13: Pregled sponzora projekta u detaljnom prikazu projekta

Uz svaki zapis u tabličnim prikazima dostupni su i gumbi za uređivanje i brisanje zapisa. Prava na svaku od ovih operacija uređena su pojedinačnim, automatski generiranim dozvolama u sustavu.

5.4. Nedostaci i prostor za nadogradnju

Ovakav pristup u razvoju aplikacije pokazao se jednostavan i brz te autor vjeruje da uz pomoć ovog rada aplikaciju mogu nadograđivati i osobe slabije tehničke pozadine. Ipak, mogućnosti aplikacije razvijene na ovaj način ograničene su na funkcionalnosti koje nudi *RAD* programski paket – bilo kakvu napredniju nadogradnju potrebno je razviti na tradicionalan način.

Glavna ograničenja primijećena u razvoju aplikacije odnose se na nedostatak mogućnosti za preuzimanjem podataka u tabličnom obliku te nemogućnost agregiranja podataka (računanja zbrojeva, prosjeka i slično). U budućnosti bi bilo korisno nadograditi funkcionalnost korištenog programskog paketa kako bi se nadomjestila spomenuta ograničenja.

Već spomenuto ograničenje također je upravljanje dozvolama koje je u programskom paketu izvedeno na nepraktičan način. Bolje rješenje s aspekta korisničkog sučelja moglo bi se postići organizacijom dozvola u stablastu strukturu te omogućavanjem dodjele grupa dozvola, no takva nadogradnja zahtijevala bi promjenu podatkovnog modela i izradu nove vrste obrasca.

S obzirom na to da je razvijeni sustav *Flask* aplikacija, te da koristi uvriježenu *SQLAlchemy* knjižnicu, nadogradnje na sustav mogu se razvijati na tradicionalan način u okvirima koje pruža *Flask* paket neovisno o paradigmi paketa *Flask-Appbuilder*. Razvoj takvih nadogradnji, ipak, zahtjeva više resursa i znanja nego razvoj aplikacije uz ovakav *low-code* pristup te stoga nije uvijek u mogućnostima jedne studentske organizacije.

Dodatna zapreka pri korištenju konkretnog *Flask-Appbuilder* paketa, već spomenuta u potpoglavlju 5.2.4, jest loša dokumentacija paketa. Većina razreda i funkcija koje paket definira nisu opisane u njegovoj dokumentaciji te se ponekad treba referirati na kod paketa što bi volonteru s manje tehničkih znanja moglo predstavljati problem. Ipak, većina potrebnih osnovnih funkcionalnosti opisana je u dokumentaciji.

Razvijenom aplikacijom također nisu pokriveni svi zahtjevi navedeni u poglavlju 3 jer bi time aplikacija prešla opseg ovog rada. Zahtjevi za implementaciju odabrani su prema subjektivnoj procjeni prioriteta proizašloj iz iskustva rada u organizaciji i razgovora s volonterima. Cilj ovog rada među ostalim je dokumentirati postupak razvoja aplikacije kako bi se omogućila buduća nadogradnja ukoliko se za tim pokaže potreba.

6. Zaključak

U sklopu ovog diplomskog rada razmotren je rad jedne studentske organizacije koja organizira natjecanje s velikim brojem sudionika na primjeru *Organizacije za studentska natjecanja STEM Games* te su iz opisa njezinog djelovanja izdvojeni najvažniji zahtjevi na informacijski sustav za podršku radu organizacije. Potom je ukratko promotrena primjenjivost *ERP* rješenja otvorenog koda u odnosu na navedene zahtjeve. Na posljetku, razvijen je i dokumentiran informacijski sustav oblikovan prema zahtjevima organizacije i implementiran pomoću *RAD* programskog okvira *Flask-Appbuilder*.

U evaluaciji oba pristupa pokazalo se da *ERP* rješenje ne pokriva dio zahtjeva koji su specifični za studentsku organizaciju ovog tipa, dok s druge strane nudi mnoge nepotrebne mogućnosti koje povećavaju kompleksnost njegove uporabe. Područje u kojemu je *ERP* rješenje u velikoj prednosti jest vođenje financija udruge, s obzirom na to da se mogućnosti koje nudi ne mogu implementirati koristeći *RAD* programski okvir.

U drugim aspektima, aplikacija razvijena pomoću *RAD* programskog okvira adekvatna je i pokriva većinu organizacijskih potreba koje se svode na pohranu podataka. Proces razvoja pokazao se jednostavnim te se iz razgovora s volonterima u organizaciji dalo zaključiti da je nadogradnja razvijene aplikacije unutar mogućnosti studenta volontera. Jedan od ciljeva ovog rada bio je i olakšati takve eventualne nadogradnje dokumentirajući postupak razvoja.

Nakon predstavljanja razvijene aplikacije, voditelji timova *Udruga* iskazali su podršku uvođenju ovakvog sustava. Primijećen je nedostatak funkcionalnosti oko financijskog aspekta aplikacije, što dovodi do zaključka da pristup razvoju korišten u ovom radu ne može u potpunosti ispuniti zahtjeve vezane uz financije *Udruga*. Taj bi se nedostatak mogao nadomjestiti nadogradnjom razvijenom na klasičan način, no s obzirom na adekvatnost *ERP* modula u tom aspektu valjalo bi razmotriti korištenje isključivo toga modula paralelno uz razvijenu aplikaciju.

Primjenjivost razvijenog sustava te praktičnost njegove nadogradnje konačno će se pokazati kroz njegovo korištenje u praksi na budućim projektima *Udruga*.

LITERATURA

- Michael Bayer. Sqlalchemy. U Amy Brown i Greg Wilson, urednici, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org, 2012. URL <http://aosabook.org/en/sqlalchemy.html>.
- M. Bradford. *Modern ERP: Select, Implement, and Use Today's Advanced Business Systems*. North Carolina State University, 2014. ISBN 9781312665989. URL <https://books.google.hr/books?id=KZRqCAAAQBAJ>.
- P. J. Eby. Pep 3333 – python web server gateway interface v1.0.1, Sep 2010. URL <https://peps.python.org/pep-3333/>. Preuzeto 16. svibnja 2022.
- Kuhn R. Ferraiolo D. Role-based access controls. U *Proceedings of the 15th National Computer Security Conference*, stranice 554–563, 1992.
- Vittorio Gianni Fougatsaro. A study of open source erp systems. Magistarski rad, Blekinge Institute of Technology, 2009.
- Daniel Vas Gaspar. Flask-appbuilder, 2013. URL <https://flask-appbuilder.readthedocs.io/en/latest/>. Preuzeto 16. svibnja 2022.
- Miguel Grinberg. *Flask web development: developing web applications with python*. "O'Reilly Media, Inc.", 2018.
- Patrick J. D. Rashid M. A., Hossain L. The evolution of erp systems: A historical perspective. 2002.
- Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, i Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. U *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, stranice 171–178, 2020. doi: 10.1109/SEAA51224.2020.00036.

Dodatak A

Struktura direktorija razvijene aplikacije

```
README.rst
__pycache__
app
app.db
babel
config.py
run.py

./app:
__init__.py
__pycache__
index.py
models.py
models.py.html
models_for_print.html
sec.py
sec_views.py
templates
translations
views.py

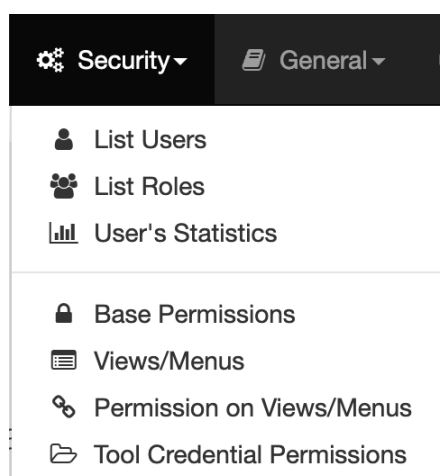
./app/templates:
404.html
my_index.html

./app/translations:
pt
```

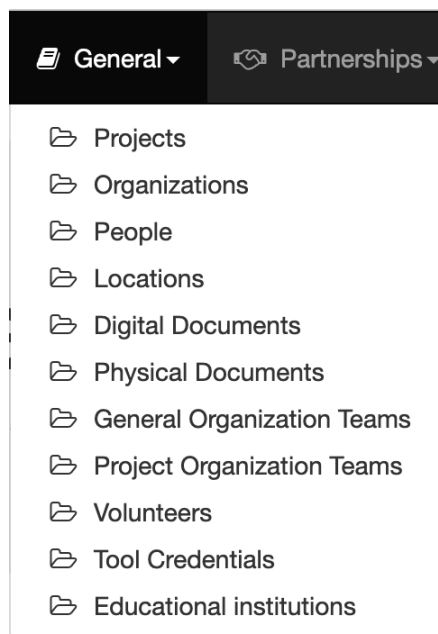
```
./app/translations/pt:  
LC_MESSAGES  
  
./app/translations/pt/LC_MESSAGES:  
messages.mo  
messages.po  
  
./babel:  
babel.cfg  
messages.pot
```

Dodatak B

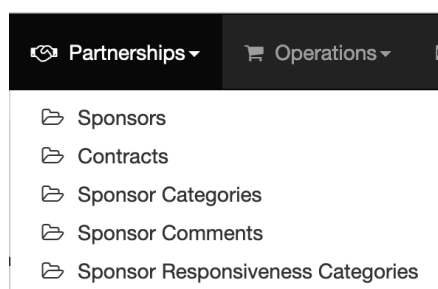
Izbornici u razvijenoj aplikaciji



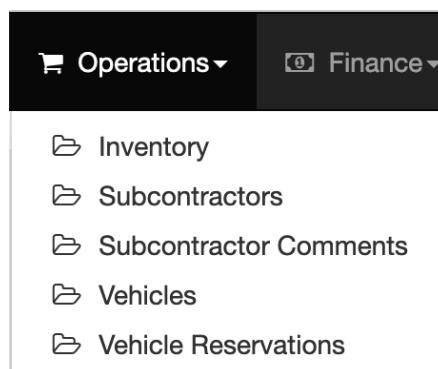
Slika B.1: Izbornik *Security*



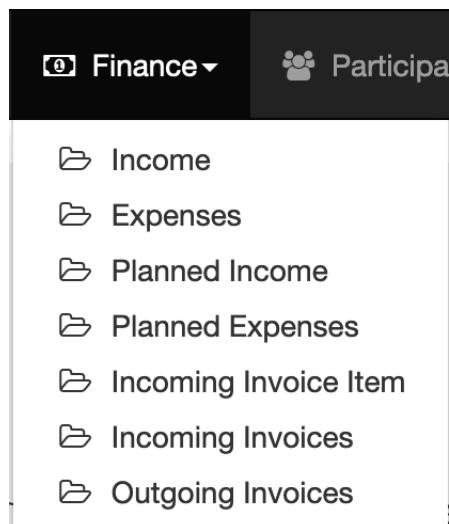
Slika B.2: Izbornik *General*



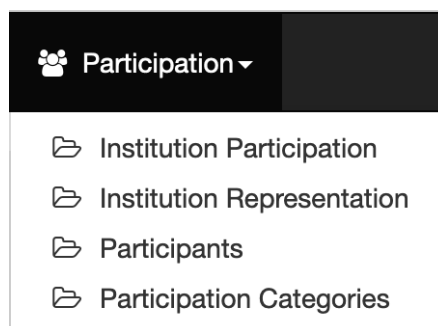
Slika B.3: Izbornik *Partnerships*



Slika B.4: Izbornik *Operations*



Slika B.5: Izbornik *Finance*



Slika B.6: Izbornik *Participation*

Razvoj informacijskog sustava za potporu organizaciji događanja s velikim brojem sudionika korištenjem programskog okvira za ubrzani razvoj aplikacija

Sažetak

Ovaj rad proučava pristup u implementaciji informacijskog sustava za potporu studentskoj neprofitnoj udruzi koja organizira studentska natjecanja s velikim brojem sudionika. Udruga tog tipa nema resursa za razvoj kompleksnog informacijskog sustava, pa su stoga promotreni nezahtjevni pristupi za njegovo ostvarenje. Zahtjevi na takav sustav proučeni su na primjeru Udruge za studentska natjecanja STEM Games te su prema njima razmotreni pristupi korištenja gotovog ERP rješenja te razvoja sustava koristeći Flask-Appbuilder programski okvir za ubrzani razvoj aplikacija (RAD). Razvijen je informacijski sustav za navedenu udruhu te je dokumentiran postupak razvoja. Pokazalo se da ERP sustav uvodi nepotrebnu složenost i ne pokriva zahtjeve specifične za studentsku organizaciju te da se korištenjem RAD okvira može brzo i lako razviti ograničen informacijski sustav koji zadovoljava specifične potrebe organizacije.

Ključne riječi: RAD, studentska organizacija, ERP, informacijski sustav

Development of an Information System to Facilitate the Organization of Large-Scale Events Using a Rapid Application Development Framework

Abstract

This thesis studies approaches in implementing an information system to support a student non-profit association that organizes large-scale student competitions. Such an association lacks the necessary resources for the development of a complex information system, so the analysis focuses on non-demanding approaches of implementation. The requirements for such a system are studied on the example of the Association for Student Competitions STEM Games and are subsequently used to assess the feasibility of the use of a ready-made ERP solution and the implementation of a custom system using the Flask-Appbuilder rapid application development framework (RAD). An information system is developed for the aforementioned association and the development process is documented. Analysis shows that the use of an ERP system introduces unnecessary complexity and does not cover the application-specific requirements, while a RAD based approach allows for easy and fast development of an information system that suits the particular organizational requirements.

Keywords: RAD, student organization, ERP, information system